The Dissertation Committee for Taewan Kim
certifies that this is the approved version of the following dissertation:

Robust Deep Fusion Models for Self-driving Cars

Committee:

Joydeep Ghosh, Supervisor

Constantine Caramanis

Haris Vikalo

Alexandros G. Dimakis

Oluwasanmi Koyejo

Robust Deep Fusion Models for Self-driving Cars

by

Taewan Kim

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2019

Dedicated to my parents and my wife

Acknowledgments

I would like to extend my deepest gratitude to my advisor, Prof. Joydeep Ghosh, for the support and guidance he has provided me throughout the long and arduous journey of graduate school. His constant trust and support helped me develop independent research problems with sufficient time for studying fundamental knowledge. He always encouraged me to learn cutting-edge techniques, investigate problems related to both the academic and real world, and improve essential communication skills as a researcher. I am sincerely grateful for his unwavering care about my happiness. In addition, I am much beholden to my committee members, Prof. Constantine Caramanis, Prof. Haris Vikalo, Prof. Alexandros G. Dimakis, and Prof. Oluwasanmi Koyejo, for their insightful comments and suggestions.

I would also like to thank Prof. Sae-Young Chung who was my bachelor's degree advisor. He gave me a valuable opportunity to experience how to do research independently for the first time. I have also had the privilege of working with Prof. Yung Yi at KAIST, who taught me right attitudes of a good researcher. I was also fortunate to work with Prof. Emily Mower Provost during my master's degree study, and her guidance expanded my horizons and helped me better understand machine learning. I believe that the numerous exceptional courses I took are the driving forces behind my research, and hence

thank Prof. Sujay Sanghavi, Prof. Sanjay Shakkottai, Prof. Evdokia Nikolova, Prof. Philipp Krähenbühl, Prof. Demosthenis Teneketzis, Prof. Honglak Lee, Prof. Alfred Hero, Prof. Raj Rao Nadakuditi, and Prof. Silvio Savarese for their invaluable instruction. In particular, I wish to appreciate Prof. Demosthenis Teneketzis for encouraging me to believe my ability and begin this journey.

My friends have been an important source of my life as a graduate student. First, I would like to thank Prof. Kangwook Lee, who introduced me to the field of machine learning, for being a friend, a mentor, and a respectable researcher. I also want to thank the members of IDEAL Avro, Shalmali, Jette, Alan, Diego, Woody, Dany, Shubham, Rahi, and Farzan, for our fun memories and Rajiv, Suriya, Ayan, Bayzid, Joyce, Yubin, and Ashish for giving me an academic inspiration. In particular, I thank Michael for his passionate contribution to our projects and valuable instruction in English. I have spent many days with my CAR-STOP colleagues Junil, Gilwon, Patrícia, Pragun, Felipe, and Saharsh. I appreciate their immense efforts to make a big step forward for the project. I am thankful to my friends in WNCG—Dohyung, Sungwoo, Namyoon, Jeonghun, Junse, Chang-sik, Jinseok, Muryong, Junmo, Kiyeon, Jeong Yeol, Vatsal, Erik, Murat, Shanshan, Ioannis, Megas, and many others—who gave me a positive motivation every day.

Finally, my gratitude goes to my family. I thank my parents for their constant support and unconditioned love. And last but not least, to Soohyun, my partner in life, thank you for always being there for me.

Robust Deep Fusion Models for Self-driving Cars

Publication No. _____

Taewan Kim, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Joydeep Ghosh

Deep learning algorithms have been adopted to various applications like self-driving cars and healthcare for their superb performance. In such fields, trustworthy models are indispensable to practical systems because their decisions are directly connected to our lives. Utilizing multiple input sources is an effective and natural way of improving a deep model's ability and robustness, because both *complementary* and *shared* information can be extracted from different sensors. In this dissertation, we focus on developing *deep fusion models* for a self-driving car's perception system.

First, a novel deep sensor-fusion convolutional neural network (CNN) architecture for detecting road users is introduced to make the system robust against natural perturbation. A laser based sensor LIDAR, which stands for Light Detection and Ranging, is selected as another input source to supplement the shortcomings of an RGB camera. Additional object proposals lead the detector to attain higher accuracies in finding and locating road users like

cars, pedestrians, and cyclists. Our algorithm further benefits from LIDAR's advantage and shows improved robustness against different lighting conditions.

Next, we develop a CNN-based pedestrian detection model which provides an additional functionality of depth prediction. The proposed algorithm learns a joint feature representation by extracting information from both RGB and LIDAR data to overcome inherent limitations of a single sensor framework, i.e. no depth information in an RGB image. Our simplified task and a direct fusion strategy make the model predict in real-time. We then introduce a newly collected pedestrian detection dataset with distinctive characteristics to test our architecture.

Finally, we investigate learning fusion algorithms that are robust against noise added to a single source. We first demonstrate that robustness against corruption in a single source is not guaranteed in a linear fusion model. Motivated by this discovery, two possible approaches are proposed to increase robustness: a carefully designed loss with corresponding training algorithms for deep fusion models, and a simple convolutional fusion layer that has a structural advantage in dealing with noise. Experimental results show that both training algorithms and our fusion layer make a deep fusion-based 3D object detector robust against noise applied to a single source, while preserving the original performance on clean data.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep learning models have accomplished superior performance in several machine learning problems (LeCun et al., 2015) including object recognition (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016; Huang et al., 2017), object detection (Ren et al., 2015; He et al., 2017; Dai et al., 2016; Redmon et al., 2016; Liu et al., 2016; Redmon and Farhadi, 2017) and speech recognition (Hinton et al., 2012; Graves et al., 2013; Sainath et al., 2013; Chorowski et al., 2015; Chan et al., 2016; Chiu et al., 2018), which use either visual or audio sources. In particular, convolutional neural networks have been actively used in visual tasks after a monumental achievement: AlexNet designed by Krizhevsky et al. (2012) outperformed all the previous algorithms for an image classification problem, a fundamental computer vision task to classify the label of a given image, with a huge gap.

One natural way of improving a model's ability is to utilize multiple input sources for a plentiful supply of information related to target variables. Formally, if we have multiple input sources $X_1, \cdots, X_m$ and a target variable

Y, mutual information can be maximized by incorporating all the inputs:

$$\begin{aligned} I(Y; X_1, \cdots, X_m) &= I(Y; X_1) + I(Y; X_2, \cdots, X_m | X_1) \\ &\geq I(Y; X_i) \quad (\forall i \in [m]) \end{aligned} \tag{1.1}$$

A combined approach *deep fusion model* is motivated by these two favorable methodologies, deep learning and fusion of multiple sources, and it has attracted considerable attention for real-world applications like autonomous driving (Kim and Ghosh, 2016; Chen et al., 2017; Qi et al., 2018; Ku et al., 2018), medical imaging (Kiros et al., 2014; Wu et al., 2013; Simonovsky et al., 2016; Liu et al., 2015), and audio-visual speech recognition (Huang and Kingsbury, 2013; Mroueh et al., 2015; Sui et al., 2015; Chung et al., 2017). In this dissertation, deep fusion models are researched with emphasis on self-driving related technologies like object detection with CNNs. However, we believe that our discoveries and insight can be easily extended to other fields using multi-modal fusion approaches.

1.1 Robust Fusion Models for Self-driving Cars

Self-driving cars must detect surrounding objects to avoid potential threats and plan paths accordingly. This perception system can rely on different types of sensors from an RGB camera to range sensors like radar and LIDAR (Light Detection and Ranging). RGB cameras have been popular in perception technologies because of relatively low price, detailed visual information similar to captured by human eyes, and powerful performance together with advanced

CNN architectures. Robust perception systems often rely on RGB image data processed by machine learning models. In fact, advanced vehicles of companies like Waymo[1] and Uber[2] are equipped with multiple cameras to fully extract surrounding visual information and make trustworthy decisions using computer vision algorithms.

Relying only on a single type of sensor may limit the ability of models and create unavoidable faulty cases. Although RGB images are widely used in object detection with deep learning, natural conditions like lighting, weather, and other image quality factors can degrade the performance of detection. Furthermore, the distance from an ego vehicle to the detected object cannot be well estimated with a single camera. A successful combination of complementary sensors can lead to a system robust against such natural perturbations and limitations. The aforementioned simple information theoretic fact about the benefit of fusion models is inherent in self-driving cars technologies. For example, Tesla utilizes 8 surround cameras, 12 ultrasonic sensors, and a forward-facing radar for their Autopilot technology[3].

However, there is no obvious solution to building a robust model using multiple input sources. Moreover, even a single unexpected behavior of the system can cause a catastrophic loss which actually occurred in 2018, a fatal crash of Uber's self-driving test car. Although cause of the accident turned out

[1] https://waymo.com/tech/
[2] https://www.uber.com/info/atg/
[3] https://www.tesla.com/autopilot

to be a complicated mixture of various issues, a report of National Transportation Safety Board (NTSB) reveals that the system also showed some faulty detection results. LIDAR and radar observations detected the victim about 6 seconds before the crash, but the pedestrian was misclassified as an unknown object, a vehicle, and a bicycle at various times[4]. This fatal example shows the importance of serious research on fusion methodologies and this dissertation studies how to strengthen deep fusion models' ability and robustness regarding the two expected benefits of fusion approaches.

1.2 Complementary Information in Fusion Models

The first advantage of using fusion models is having access to information with increased diversity. Different sensors can contribute to the desired prediction in various ways for a given specific task, and we call this source-specific knowledge *complementary information* throughout this work. For example, a single RGB camera cannot provide absolute distances of any observed objects, whereas range sensors like radar and LIDAR directly output depth information. On the other hand, color is a unique feature which can be only provided by an RGB camera. Therefore, fusing all the sensors should improve the performance of a model if a target variable to predict is dependent on both color and depth information of an observed scene. In recent years, improving robustness of a self-driving car's detection system with deep fusion models utilizing these

[4] https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf

complementary sensors has been a core technology and an active research problem (Feng et al., 2019).

1.3 Shared Information in Fusion Models

Shared information is another benefit of taking fusion approaches for improving a model's robustness. Similar knowledge contained in different types of sensors can be extracted either manually with predefined processes or automatically by deep learning models. Specifically, shared information helps when the information to each sensor is randomly erroneous. As a simple example, if all the input sources provide same information with different errors, a classical fusion strategy like ensemble methods can reduce the error variance (Tumer and Ghosh, 1996a,b). Similarly, existence of shared information in multiple sensors may help improving robustness against corruption in the subset of input sources. For this reason, a LIDAR sensor is used in self-driving technologies as it can sense the existence of any objects during night time when RGB cameras perform defectively. However, research on effective methods to utilize shared information have attracted less attention than improving a model's performance with complementary features.

1.4 Dissertation Contributions

Developing trustworthy models is essential for deployment of self-driving cars, due to their direct and crucial impacts on our lives. Thus, this dissertation focus on effective ways of utilizing multiple input sources based on deep learning

models for better performance and improved robustness. Our deep fusion models and corresponding algorithms are invented with careful consideration of CNN frameworks, characteristics of sensors, and possible vulnerabilities.

Classical late fusion methods learn multiple models separately and combine all predictions afterward. Suppose there exist n_s input sources $x_1, \cdots x_{n_s}$ and a target variable y to predict, late fusion approaches like ensemble methods learn n_s different models f_i's to estimate $p(y|x_i)$ and get a final prediction as an average of them (Figure 1.1). This solution is known to be an optimal Bayesian decision rule if the following two conditions hold (Kittler et al., 1998):

$$(1) \quad p(x_1, \cdots, x_{n_s}|y) = \prod_{i=1}^{n_s} p(x_i|y)$$
$$(2) \quad p(y|x_i) = p(y)(1+\delta_i) \quad (\delta_i \ll 1)$$

The first condition represents conditional independence, and the second one assumes that the posteriori probabilities are similar to the prior probabilities. However, it might not be easy for these conditions to be satisfied in reality. Furthermore, an optimal solution we want to estimate is $p(y|x_1, \cdots, x_{n_s})$ not an average of $p(y|x_i)$'s.

One of the huge advantages of using deep learning models is an automatic feature extraction from raw data. Therefore, our deep fusion models rather assume existence of an ideal latent representation z which satisfies the following condition:

$$p(y|x_1, \cdots, x_{n_s}) \approx p(y|z)$$

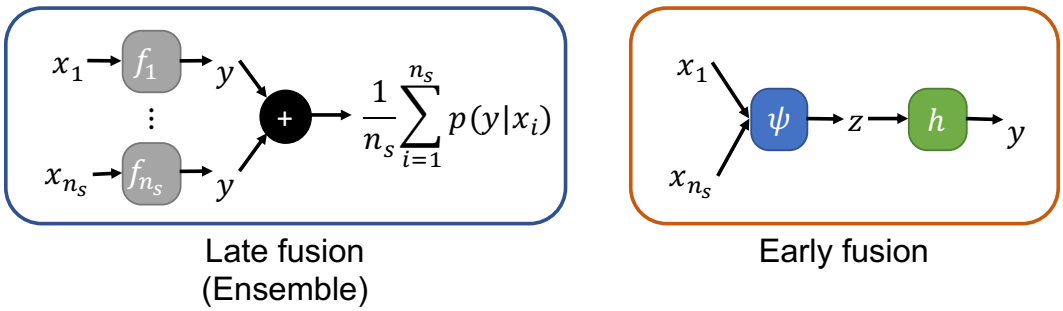


Figure 1.1: Different fusion methods, late and early

We put more effort on developing algorithms with a strategy similar to an early fusion method, which is illustrated at the right side of Figure 1.1.

1.4.1 Robust Road User Detection with Deep Fusion

Our first contribution is a multi-sensor algorithm to detect non-motorized road users, such as cyclists and pedestrians. This is a challenging problem in collision warning/collision avoidance (CW/CA) systems as direct information (e.g. location, speed, and class) cannot be obtained from such users. Thus, we propose a fusion of LIDAR data and a deep learning-based computer vision algorithm, to substantially improve the detection of region proposals and subsequent identification of road users. Experimental results on the KITTI object detection benchmark (Geiger et al., 2012) quantify the effectiveness of incorporating LIDAR data with region-based deep convolutional networks.

1.4.2 Object Detection with Simplified Depth Prediction

Though RGB Cameras, Radar and LIDAR are popular sensors for self-driving cars, real-time joint inference on their sensory outputs remains challenging. Moreover, high-resolution LIDAR is expensive both in terms of cost and computation. Our second contribution presents a deep learning-based pedestrian detection algorithm that takes both RGB images and lower-resolution LIDAR data and returns object detections in the image as 2D bounding boxes, plus the distances of the detected objects. The proposed network is much less expensive but comparable in accuracy to previous deep networks that combine these sensors use image-like or voxel representations of LIDAR data to directly predict 3D positions and shapes. To train this network, a new dataset is created, containing register information from low-end camera to a 16-layer LIDAR, and corresponding ground truth distance values generated by estimating the position of pedestrians from global navigation satellite system (GNSS) sensors and a fixed tower. The public release of this dataset is an additional contribution of this effort.

1.4.3 Robustness Against Single Source Faults

The final contribution studies a fundamental problem of vulnerabilities in fusion models. Although it is reasonable to expect that existence of shared information in multiple input sources should make a fusion model robust, we show that additional strategies are still required to obtain robustness against noise or corruption on a single source. We first provide analytical results based

on a simple data model to simulate characteristics of fusion models. Inspired by the analysis, training algorithms for our novel loss and an effective fusion layer are proposed to support our findings with experimental results on more complex and practical settings using deep fusion based 3D object detectors.

1.4.4 Organization

This dissertation is organized as follows. Chapter 2 examines deep learning algorithms for computer vision tasks focusing on object detection, and characteristics of different sensors used in self-driving cars are described. The main sensors used in our dissertation are an RGB camera and a rotational LIDAR, and we further discuss about object detection algorithms using both sensors. Chapter 3 unveils our first deep fusion based road user detection algorithm, which shows robustness against natural perturbation like lighting conditions. Chapter 4 proposes another type of deep fusion model with a single-stage architecture, which can jointly detect pedestrians and predict depth information in real-time. Then we introduce two novel methods to achieve robustness against corruption in a single input source, which are supported by analyses on a simple model and empirical validation with deep complex models. We conclude the dissertation in the last chapter by providing some directions for interesting future work about robustness against adversarial attacks and interpretation of fusion models.

Chapter 2

Background

2.1 Object Detection and Evaluation

There are three fundamental problems in computer vision which have been actively tackled by deep learning methods in recent years, (i) image classification, (ii) object detection, and (iii) semantic segmentation. For a perception system of a self-driving cars, object detection is one of the key tasks as understanding of surrounding objects is the crucial for planning safe paths. Not only road users like pedestrians, cars, cyclists, and motorcycles but also some stationary objects like traffic lights and signs can be detected with object detection algorithms. Therefore, we focus on developing deep fusion models for object detection[1]. Figure 2.1 illustrates differences among the three computer vision problems where a predefined set of labels is $\mathcal{Y} = \{\text{Car}, \text{Person}\}$.

Definition 2.1. ***Image classification*** *is the problem of predicting a class label $y \in \mathcal{Y}$ for a given image input x, where $\mathcal{Y}$ is a predefined set of classes. The number of true label per image is exactly one.*

[1]Segmentation is another important problem for self-driving cars and recent deep learning models like Mask R-CNN (He et al., 2017) show great performance on image data.

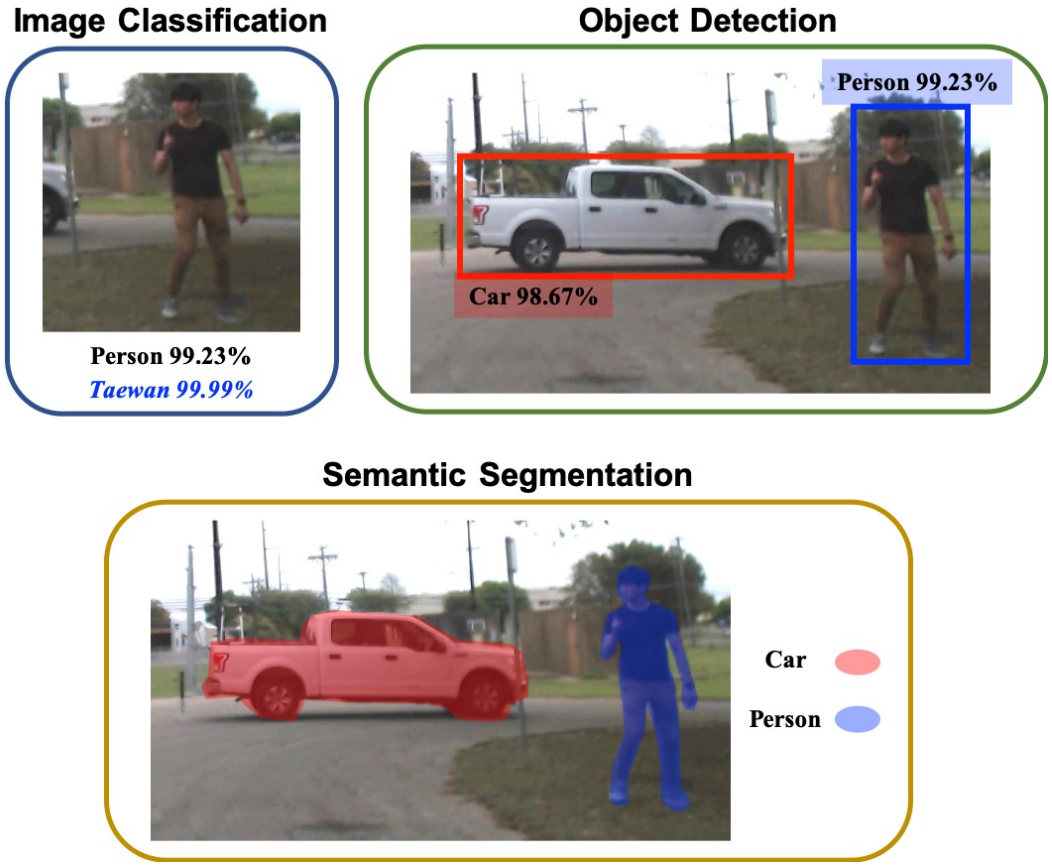


Figure 2.1: Different problems in computer vision

Definition 2.2. ***Object detection*** *is the problem of finding all objects in a given image x, where the detected objects are labeled with a predefined set of classes $\mathcal{Y}$. A model should provide not only the class of the detected object, but also coordinates of a corresponding bounding box which locate the object in an image.*

Definition 2.3. ***Semantic segmentation*** *is the problem of finding meaningful parts in a given image x, where the segmented parts can be labeled with a*

predefined set of classes $\mathcal{Y}$. A model should label each pixel with the $|\mathcal{Y}|$ classes.

Intersection Over Union

In object detection, evaluating qualities of bounding boxes is required to correctly locate target objects in a 2D or a 3D space. For a given ground truth box and a predicted bounding box, a metric called intersection over union (IOU) quantifies the similarity between them. In detail, IOU calculates the ratio of two areas, intersection over union of the boxes. Figure 2.2 depicts IOU of two boxes.

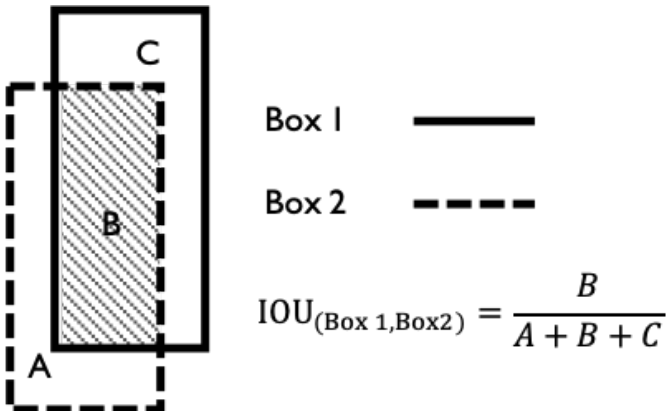


Figure 2.2: Visual illustration of IOU calculated from two overlapping boxes.

IOU of a ground truth box and a predicted box is the principal both in training and evaluating the performance object detection models. First, numerous candidate boxes can be provided for a single target object while training a model. Then we have to decide which boxes to include as positive samples so that useful gradients are computed from a loss function to update parameters. For evaluating a detector's performance, both the predicted class

and the region of a box should match with a ground truth object. In general, IOU with the target box should exceed a certain threshold to ensure the quality of a predicted object's bounding box, e.g. IOU score must be at least 50% to be declared as a true positive for classes like pedestrian and cyclist. Figure 2.3 visually shows an example of a true positive detection sample.

Figure 2.3: An example of a positive sample, i.e. a correctly detected object, defined based on IOU of a ground truth box and a predicted box.

Average Precision

Average Precision (AP) is one of the standard evaluation metrics in object detection. After obtaining numerous detected boxes with corresponding class label predictions, each bounding box can be classified into four different cases based on the fixed IOU threshold score and the varying probability score thresholds for the class: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Then two standard evaluation metrics *precision*

and *recall* can be calculated per each class label.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Conceptually, AP calculates the area under the precision-recall curve to measure how precision values change over the recall scores, which can be determined by the threshold score for the predicted class probability. In practice, AP score is approximated by computing mean of precisions at a set of equally spaced recall levels (details described in Everingham et al. (2010)). In this dissertation, a threshold value for IOU can be specified as AP@0.5, if all the detections with IOU$\geq$ 0.5 are considered as candidates for positive samples.

2.2 Object Detection with Deep Learning

In the past few years, deep learning has been applied to multiple state-of-the-art algorithms for object detection in image data. Object detection algorithms using CNNs can be classified into two categories based on their frameworks.

2.2.1 Region-based object detection

The first type of these techniques, Region-based Convolutional Neural Networks (R-CNN) (Girshick et al., 2014), introduced an effective way of utilizing deep learning algorithms in object detection, which substituted the traditional sliding window approach and other feature extraction methods including SIFT (Scale-Invariant Feature Transform) (Lowe, 2004), HOG

(Histogram of Gradients) (Dalal and Triggs, 2005), edgelet (Wu and Nevatia, 2005), shapelet (Sabzmeydani and Mori, 2007), and others. It incorporates a separate region proposal algorithm, such as Selective Search (Uijlings et al., 2013), to provide rich and high quality object candidates called *regions of interest (ROIs)* to image classification modules. SPP-net (Spatial Pyramid Pooling) (He et al., 2014) and Fast R-CNN (Girshick, 2015) improved the speed of R-CNN by reducing the calculation of convolutional features. Faster R-CNN (Ren et al., 2015) introduced a small convolutional network, Region Proposal Network (RPN), to further speed up. A framework of Faster R-CNN has become a standard deep learning method for object detection, and key steps of the algorithm are as follows:

(1) Extract features using a base CNN from the input image.

(2) At each sliding-window or location of the extracted feature, generate anchor boxes with predefined shapes (scales and aspect ratios).

(3) RPN predicts the probability of a box having an object or not (two-class classification) per anchor box, and regress the box coordinates to fit the object (bounding box regression).

(4) Object proposals (CNN feature maps) with different sizes are transformed to fixed size feature maps (ROI pooling).

(5) The feature maps are provided to convolutional layers to predict their classes and regress the bounding boxes.

(6) If multiple boxes are predicted for a single object, an algorithm called *non-maximum suppression* (NMS) eliminates some overlapping candidates based on their probability scores.

Although these methods improved both quality and speed of the object detection algorithms, their inference speed was not enough for robust real-time applications like autonomous driving. Recently, Mask R-CNN was introduced by He et al. (2017) with additional functionality of predicting object masks, but the focus was more on the instance segmentation not the speed. Our first deep fusion model introduced in Chapter 3 modifies the process of Fast R-CNN for robust detection. Also, a baseline algorithm used in Chapter 5, AVOD (Ku et al., 2018), incorporates RPN and fusion strategies for 3D object detection.

2.2.2 Single stage object detection

Single stage detection algorithms improve inference speed by avoiding a separate region proposal step. These methods does not use a second network like RPN for region proposal but do both class prediction and box regression with only a single network. You Only Look Once (YOLO) (Redmon et al., 2016), one of the most popular single stage detectors, predicts a fixed number of bounding boxes with confidence score from each divided grid of an image. Improved versions, YOLO v2 (Redmon and Farhadi, 2017) and v3 (Redmon and Farhadi, 2018), show powerful detection performance running in real-time ($\geq$40fps). Single Shot MultiBox Detector (SSD) is another state-of-the art single shot method (Liu et al., 2016): the convolutional network provides feature

maps from multiple network stages to handle different scales, and bounding boxes and probability scores for each classes are predicted simultaneously. Our second deep fusion architecture introduced in Chapter 4 extends the original SSD for an RGB image to the model using multiple sources.

Independent from structures of object detection algorithms, one of the most recent works tackled the problem of having an extreme imbalance between foreground and background classes during training in single shot detections by developing a loss called *Focal Loss* (Lin et al., 2017).

2.3 Sensors for Self-driving Cars

It is easy to get news about self-driving cars testing around different cities successfully, and numerous companies and research groups have dived into the field of autonomous vehicles. However, there is no standard rule for setting up sensors including which types of sensors to use. In fact, it is still debatable whether using LIDAR is necessary for self-driving technologies. For example, Elon Musk, a co-founder of Tesla, stated that *"Anyone relying on LIDAR is doomed."* at the Tesla's Investor Day event[2]. On the other side, leading companies in autonomous vehicles like Uber, Waymo, and automakers use LIDAR to capture rich information in 3D.

Imaging devices or cameras are the basic sensors for self-driving cars. The first advantage of using a camera is the form of data given to users. Visual

[2]An article of TechCrunch posted on April 22, 2019. https://techcrunch.com/2019/04/22/anyone-relying-on-lidar-is-doomed-elon-musk-says/

representation of surrounding environment provides detailed information about the objects of interest which coincides with human eyes' perception. Thus, large number of successful researches have been accomplished in computer vision. Furthermore, price of a single sensor is relatively inexpensive compared to other sensors like LIDAR. Therefore, installing multiple cameras to cover 360-degree view is a popular solution in the field. However, there are some known limitations of using cameras. First, sensing quality becomes low when visibility condition is poor, e.g. foggy or rainy days, under shadows, and during nighttime. Also, 3D information like distances or angles cannot be exactly estimated unless some carefully designed advanced algorithms are applied with multiple cameras.

Radar is another common automotive sensor which can estimate both location and speed of an object by analyzing the reflected signals of radio waves. Therefore, it can provide an object's distance and speed information even in low visibility conditions. For this reason, a long-range radar is usually located to cover frontal region of the vehicle, and more radars can be located to detect other sides. However, it is not easy to distinguish various types of objects on the road with radar signals.

LIDAR is one of the hottest sensors in autonomous driving technologies for its special sensing data. Instead of radio signals, LIDAR uses lasers to estimate distances from the detected objects, and hence, robust against low

illumination conditions. Velodyne[3] developed a rotational sensor with multiple lasers to provide a dense 3D map in 360 degree. This rich information make it possible for the system to understand surrounding environments in real-time. However, a price of the sensor is comparatively expensive (thousands of dollars), so ongoing efforts on reducing the price can be found in many companies and research teams. Also, laser's sensing performance drops when emitted lights can be affected, e.g. during heavy rain or foggy days, on reflective surfaces, and in longer ranges.

Other types of sensors also support self-driving cars in different ways. Inexpensive ultrasonic sensors can cover short ranges to detect nearby cars or support parking. GPS can help localize the vehicle and help planning the paths in real-time. Furthermore, wireless communication sensors like Dedicated Short-range Communications (DSRC) or 5G devices are considered to communicate with intelligent transportation systems (ITS). Traffic information gathered from ITS contributes efficient path planning or recognizing some emergencies in real-time.

A vehicle used for our data collection and location of different sensors in the car is illustrated in Figure 2.4. For the front-facing camera, Logitech's HD (1280×720) webcam is installed below the car's rear view mirror. And the LIDAR (VLP-16[4]) is mounted on the roof with our customized metal box.

[3] https://velodynelidar.com/
[4] https://velodynelidar.com/vlp-16.html

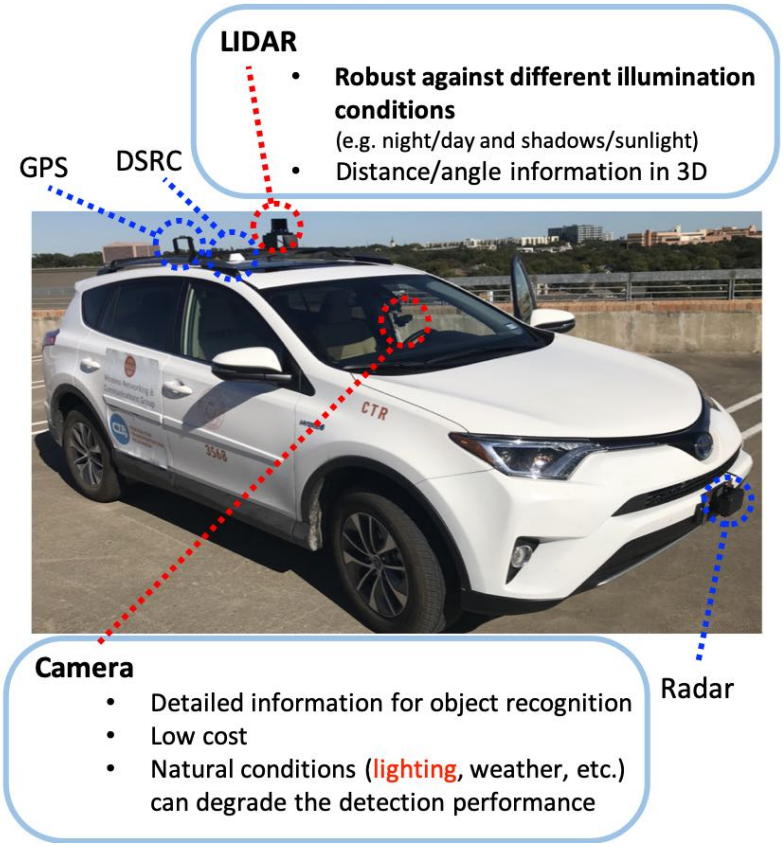
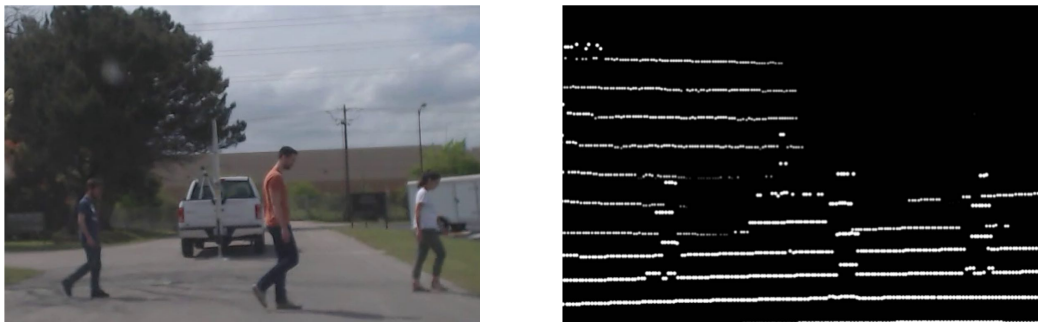


Figure 2.4: Different sensors used in intelligent/autonomous vehicles

2.4 Fusion of LIDAR and RGB Camera

Raw LIDAR data is composed of point clouds where (x, y, z) measurements are provided with additional values like intensity and reflectance. One way of providing such data into CNN architecture is to project 3D points onto 2D image space. An example of a LIDAR image and a corresponding RGB image is provided in Figure 2.5 which is sampled from our new dataset collected for the algorithm to be introduced in Chapter 4. The dataset uses a lower-resolution LIDAR sensor, specifically, the 16-layer VLP-16, whereas the

KITTI dataset provides a 64-layer point clouds data. Low resolution LIDAR provide vertically sparse information, so techniques designed for high resolution LIDAR require careful adaptation.



(a) Detections from tower (b) Detections mapped to ground

Figure 2.5: Examples of a RGB image and a corresponding LIDAR data projected to the image space

Purely LIDAR-based detection algorithms have attracted increasing interest from researchers (Engelcke et al., 2017; Asvadi et al., 2017; Zhou and Tuzel, 2017; Saleh et al., 2017; Matti et al., 2017; Lang et al., 2019; Shi et al., 2019), but sensor-fusion approaches using both RGB image and LIDAR point clouds can more easily classify objects and modify existing network architectures. Some methods extend Fast R-CNN by providing more object proposals from different sensors (Braun et al., 2016) including our first work (Kim and Ghosh, 2016). However, these methods do not run in real-time and not directly incorporate LIDAR into detection network. MV3D (Multi-View 3D network) algorithm introduced a multimodal fusion-based deep learning algorithm which has 3 inputs: (a) RGB image, (b) LIDAR front view, and (c) LIDAR bird's eye view (BEV) (Chen et al., 2017). Different fusion schemes,

early/late/deep fusion, methods in deep learning architectures are also analyzed by the authors. This approach requires LIDAR of a high enough resolution to create a detailed front view "image".

Recent deep fusion methods have introduced a continuous fusion scheme for easier fusion of RGB and LIDAR's BEV images at different levels of resolution (Liang et al., 2018). This framework was improved in MMF (Liang et al., 2019) by training a model for multiple tasks for better feature representation. AVOD (Ku et al., 2018), a recent open-sourced 3D object detector, generates region proposals from RPN using RGB and LIDAR's BEV images. For extensive literature reviews, please refer to the recent survey papers about deep multi-modal learning methods in general (Ramachandram and Taylor, 2017) and for autonomous driving (Feng et al., 2019).

Chapter 3

Robust Road User Detection with Deep Fusion

Research in collision warning and collision avoidance (CW/CA) systems for non-motorized road users has attracted increasing interest from both automotive industries and transportation safety departments. An intelligent transportation system, which can automatically detect cyclists and pedestrians, is essential for protecting vulnerable road users. Developing a reliable detection algorithm is a starting point for devising systems that can provide information about potential collisions with pedestrians and cyclists to drivers. Despite the growth of such technologies, developing CW/CA algorithms of non-motorized road users is a challenging problem compared to detection of motorized users (cars, trucks, etc.) due to two critical differences. First, physical characteristics of cyclists and pedestrians are dissimilar to that of other road users such as cars, trucks and motorcycles. Pedestrians and cyclists have both lower range of speed and acceleration and smaller sizes. Furthermore, each type of user

This chapter has been published as: Taewan Kim, Joydeep Ghosh, "Robust Detection of Non-motorized Road Users using Deep Learning on Optical and LIDAR Data", IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, Rio de Janeiro, Brazil. The author of the dissertation contributed to problem formulation, model development, implementation, and empirical evaluation.

is endowed with different degrees of technology. It is not reasonable, at least at the moment, to assume that pedestrians and cyclists are equipped with transmitting devices, radar, speedometer, etc. Therefore, CW/CA algorithms on a vehicle should be developed without relying on direct information from non-motorized road users.

This chapter focuses on the second difference and hence on the types of sensors (and associated algorithms) that can be used to detect pedestrians and cyclists. There are three main types of sensors that can be used for assisting the interaction between automobiles and non-motorized road users. Sensors like radar and LIDAR can detect the relative position of any object/user in proximity to the vehicle, and therefore, may be the most important source of information in this situation. Communication, on the other hand, can be valuable in situations when a vehicle may detect the presence of a cyclist on the road and send this information to other neighboring vehicles. Then, the information received from other vehicles can be utilized as a information source for CW/CA algorithms.

The third on-vehicle sensor is an optical sensor such as a camera. Vision sensors are currently the most widely used sensors in developing object detection algorithms for their lower cost relative to other sensors and applicability to other tasks like lane detection. In the field of computer vision, object detection and tracking has a long history with promising results. Furthermore, there is substantial early computer vision research focused on pedestrian detection (Dollar et al., 2012; Benenson et al., 2014; Enzweiler and Gavrila,

2009; Geronimo et al., 2009) and cyclists (Cho et al., 2010; Gu and Kamijo, 2014).

Additional sensors like radar and LIDAR can be used as a supplementary source of information to enhance computer vision oriented algorithms. Fusion of radar and computer vision has also been studied before, as radar can not only provide accurate distance and angle but also has robustness against bad weather, which limits vision sensors (Milch and Behrens, 2001; Bertozzi et al., 2008; Wang et al., 2011). 3D LIDAR with high definition has recently attracted considerable attention because of its accurate depth information. Various researches have recently showed the effectiveness of combining data from LIDAR and RGB images in vision-based object detection algorithms (Cho et al., 2014; González et al., 2015; Premebida et al., 2014).

We present a simple strategy for incorporating LIDAR information with a successful computer vision oriented method for detecting pedestrians and cyclists. Deep learning-based computer vision algorithms have been heavily researched in the last few years and yielded higher performance for many visual recognition tasks. Specifically, we used one of the initiative region-based convolutional network methods, Fast R-CNN (Girshick, 2015), for object detection. After analyzing regions of interest extracted from RGB images, we have devised a simple LIDAR fusion method to support the region proposal stage of Fast R-CNN. To train and evaluate our detection algorithm, the most popular dataset called the KITTI Benchmark Suite (Geiger et al., 2012) is used. KITTI provides both scanned points from LIDAR around car and RGB

images.

The remaining parts of this chapter are organized as follows. Section 3.1 reviews challenges in non-motorized road user detection and related work on vision-based detection of pedestrians and cyclists. A brief explanation of region-based CNN algorithm and description of proposed LIDAR fusion model is described in Section 3.2. Characteristics of KITTI dataset and experimental results with analysis will be covered in Section 3.3 before the conclusion in Section 3.4.

3.1 Related Work

3.1.1 Challenges in pedestrian and cyclist detection

Computer vision approaches use images as an input to the detection algorithm. Even though a visual source includes plenty of information, pedestrian detection/tracking is still a challenging problem for several reasons. First, appearance of pedestrians exhibits high variability and requires a robust model with capability of handling various figures within the pedestrian class. Secondly, outdoor urban scenarios include some interference such as cluttered background and varying quality of sensing from weather conditions and illumination. In addition, pedestrian detection should deal with highly dynamic scenes caused by motion of pedestrian and camera, different view angles, and large distance range. Finally, application to intelligent transportation systems requires high performance in terms of both accuracy and speed.

Although the overall process is similar to pedestrian detection, cyclist

detection can be even more challenging. First, a cyclist video contains both bicycle and a human rider, and motion speed is significantly faster than the pedestrian's walking speed. High variety in the image space can cause inconsistency of features extracted from a single class, "cyclist". Further issues related to appearance changes in the cyclist detection problem are pointed out by Cho et al. (2010). First, different camera viewpoints induce dramatic variety, because shape and size appear dissimilar when viewed from the front versus the side. Class variability (e.g. mountain bikes vs. racing cycles), and appearance of the rider further compound the difficulties in recognition.

3.1.2 Computer vision for pedestrian and cyclist detection

In object detection/tracking area, computer vision researchers have focused more on pedestrian detection compared to cyclist detection. The general architecture of pedestrian detection/tracking system for an advanced driver system with computer vision methods is divided by Geronimo et al. (2009) into preprocessing, foreground segmentation, object classification, verification/refinement, tracking, and application. In this chapter, extracting regions of interest and object classification are studied with emphasis on the effectiveness of additional information from LIDAR sensors.

Extraction of ROI in foreground segmentation is an integral part of the object detection. Recently, importance of effective region proposal in object detection and comparison of popular algorithms are highlighted by Hosang et al. (2016). Traditional approaches in computer vision commonly use the sliding

window technique for providing ROIs. This technique extracts the feature from a partial rectangular region of the image and feeds it to the classifier. Then, it slides the rectangular box to process the algorithm again on different portion of the image. Since the sliding window process applies the classification algorithm on a large number of rectangular parts, computational efficiency is very low. Also, providing meaningless ROIs to the classifier can increase the possibility of erroneously identifying a pedestrian or cyclist where there is not one. Therefore, an effective algorithm for searching candidate ROIs in the image plays an important role in improving speed of detection and decreasing false positive rates.

Another important step of the object detection is the object classification phase, which includes feature extraction and learning a classifier. From the provided ROI, features are extracted so that it can produce sufficient information to recognize the target in the image. Large research efforts on finding effective features established the effectiveness of features such as Haar wavelet (Papageorgiou and Poggio, 2000), SIFT (Scale-Invariant Feature Transform) (Lowe, 2004), HOG (Histogram of Gradients) (Dalal and Triggs, 2005), edgelet (Wu and Nevatia, 2005), shapelet (Sabzmeydani and Mori, 2007), etc. These features are extracted based on pixel difference, shape, edge in object, gradients, and other calculations among close pixels to capture the characteristic shape and appearance in the image. Then, the extracted features of images become input of the classifier to learn the parameters and declare the class of object in the given ROI. One of the successful object classification strategies is the

deformable part-based model approach, which models the object into different parts and builds a classifier based on those parts (Felzenszwalb et al., 2008, 2010). It was quite a natural approach to model the human body, and these type of algorithms achieved the best performance before the advent of deep learning based models.

In recent object detection and image classification tasks, deep learning-based models with convolutional neural network (CNN) architecture have achieved state-of-the-art performance and are being widely applied. Typically, such models utilize a convolution layer to capture underlying dependencies between neighboring features. Although training and testing require a large amount of data and long computation time, developments in computation power (e.g. multi-core CPU and GPU computing) and plentiful sources of images like ImageNet (Deng et al., 2009) have accelerated the development of deep learning-based object detection. A unique advantage of the CNN approach is that it can extract effective general features from images without applying a part-based model or other complex feature extraction processes. In practice, existing pre-trained models can be applied to new object detection tasks as a feature extraction module to obtain general features capturing characteristics of the target. Results and evaluation of pedestrian detection with CNN based approach are presented by Hosang et al. (2015).

Details for the history of algorithmic development in pedestrian detection are well explained in papers like Dollar et al. (2012); Benenson et al. (2014); Enzweiler and Gavrila (2009); Geronimo et al. (2009). Also, there exists a

different type of work focused on distinguishing cyclist from pedestrians (Gu and Kamijo, 2014), which is actually an important distinction to ensure as the top part of a cyclist can be mis-identified as a pedestrian.

3.2 Deep Learning-based Algorithm with Sensor Fusion

3.2.1 Fast R-CNN for pedestrian and cyclist detection

An algorithm called Fast R-CNN is selected as a strong baseline model as we investigate the possibility of applying deep learning in real world systems with sensor fusion. R-CNN (Region-based CNN) was first proposed in Girshick et al. (2014), which combines a region proposal stage and a convolutional neural network. CNN has several well-known models that achieve state-of-the-art image classification performance (Krizhevsky et al., 2012; Girshick et al., 2014; Chatfield et al., 2014; Simonyan and Zisserman, 2015). By applying an algorithm that proposes interesting candidate regions in the image regardless of the class of the target, the number of ROIs decreases while meaningful regions remain. Since the original R-CNN was computationally expensive, improved versions were proposed later: Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015).

Fast R-CNN is composed of a separate region proposal stage and CNN stage, thus more improvable spaces exist to combine with additional sensor information; Faster R-CNN computes ROIs with deep nets by sharing convolutional layers (See Section 2.2.1 for details). In Fast R-CNN, both original image and ROIs are projected to CNN in a fixed scale to extract fixed length feature

vectors. Its outputs are fed into a classifier and a bounding-box regressor, which output class posterior probability scores and predicted bounding-box offsets per class, respectively.

3.2.2 Fusion of LIDAR data with Fast R-CNN

Well-known advantages of LIDAR are robustness against different illumination conditions (eg. night/day, and shadows/sunlight) and accurate distance/angle information in 3D. On the other hand, LIDAR is expensive, has limited range, and is inaccurate in bad weather such as heavy rain and fog. One natural fusion approach is to decrease the number of candidate ROIs extracted from the camera based on LIDAR's distance data. From statistics of depth in certain region, one can possibly declare the existence of potential objects in the ROI. This method can be devised as a module that rejects region proposals with no potential objects in order to decrease the false positive rate and improve the speed.

In Fast R-CNN, Selective Search (Uijlings et al., 2013) is chosen as a main algorithm for the region proposal stage, which produces hierarchical group of pixels based on a predefined similarity. Even though it extracts a number of effective ROIs quickly, provided ROIs include many regions with no object inside and not all objects in the image are found. Therefore, we focused on providing more high quality region proposals that are hard to be captured in RGB data. With this fusion strategy, ROIs missed in RGB data can be covered by LIDAR-based region proposals. Details of this motivation

are explained in Section 3.3 with the analysis on region proposals from RGB images. A similar sensor fusion framework for detection system is provided in Wang et al. (2011) as a 3-stage strategy with radar instead of LIDAR. The first step is to align and calibrate coordinate systems of radar and vision. Then the system searches potential targets from both sensors. Finally, it detects objects on the road from these proposed regions.

3.2.3 Camera+LIDAR Fusion Fast R-CNN

A key idea of our proposed model Camera+LIDAR Fusion Fast (CLF2) R-CNN is to utilize additional region proposals from LIDAR without harming Fast R-CNN's performance on image data. In the region proposal stage, ROIs are extracted from an RGB image with Selective Search's fast mode. Also, 3D LIDAR data is projected onto the coordinate system of the RGB image resulting in a sparse LIDAR image. Then the depth, or intensity, image is interpolated with an inpainting module provided by OpenCV (Bradski, 2000). Bounding-box locations of ROIs are extracted from this inpainted LIDAR image using Selective Search's intensity mode to deal with gray-scale images. Next, ROIs obtained from both RGB and LIDAR images are provided to CNN, so that feature vectors can be extracted from RGB image box at the location of those ROIs. Finally, a fully connected neural network layer provides class probability scores from softmax classifier and object's location from bounding-box regressor. Selective Search is used as its detection performance with R-CNN and effective region proposal quality have been proven to be one of top detection proposal

algorithms (Hosang et al., 2016). Figure 3.1 summarizes the overall steps of our CLF2 R-CNN, and five main steps are enumerated as follows:

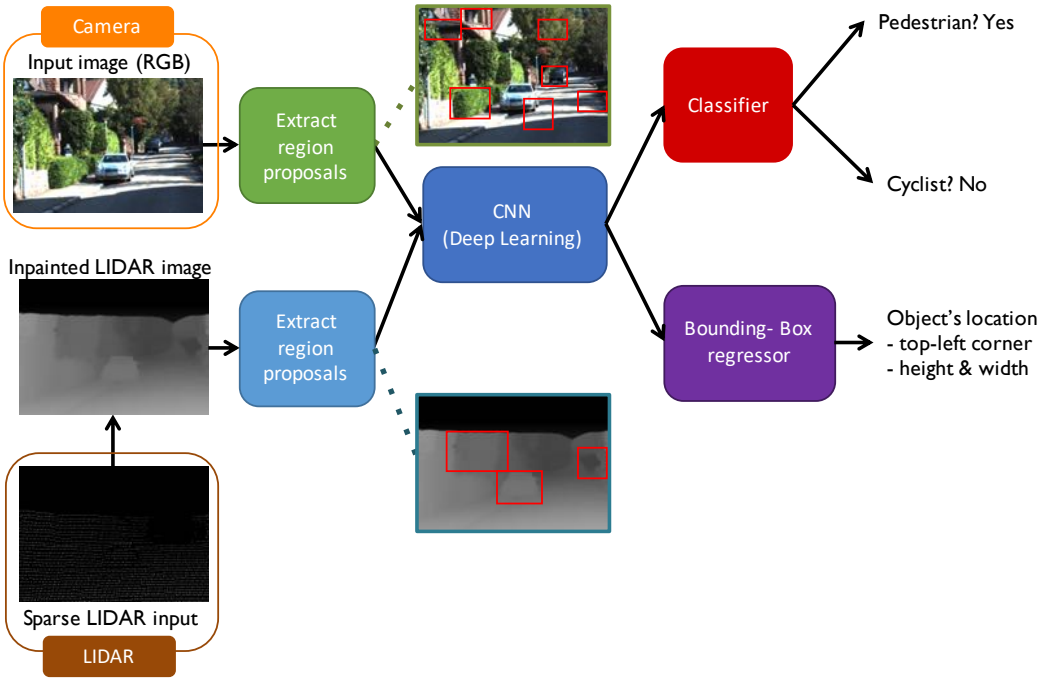


Figure 3.1: Architecture of our CLF2 R-CNN. Input data are separately collected from camera and LIDAR to extract region proposals. A CNN provides fixed length feature vectors from these region proposals to object classification and bounding-box regression units to determine class and location of objects.

(1) Project 3D LIDAR point clouds onto the 2D image space.

(2) Interpolate sparse LIDAR images. (Figure 3.2)

(3) Region proposals from both RGB and LIDAR images. (Selective Search (Uijlings et al., 2013))

(4) Extract ROIs from a convolutional feature map (RGB) using the region proposals of both RGB and LIDAR.

(5) Predict class probability scores and object's location. (Fully-connected layers)

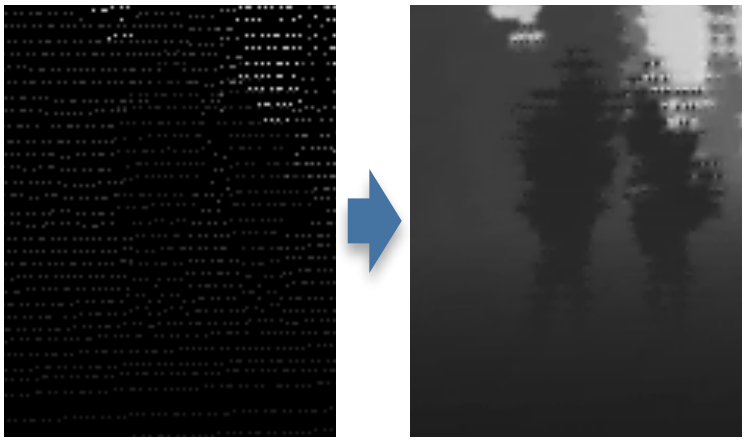


Figure 3.2: Inpainting of sparse 2D LIDAR image to dense depth image

3.3 Experimental Results

3.3.1 KITTI object detection benchmark

The most popular pedestrian detection benchmark is the Caltech Pedestrian Detection Benchmark (Dollár et al., 2009). However, it does not include labels for cyclists. Therefore, we used a more recent dataset for autonomous driving, namely, the KITTI Vision Benchmark (Geiger et al., 2012). The object detection dataset of KITTI includes 7481 training images and 7518 test images. Seven class labels are provided: car, van, truck, pedestrian, person sitting,

cyclist, and tram. For evaluation, we considered only car, pedestrian, and cyclist classes. One problem in development and evaluation of the model comes from the absence of ground truth labels in the testing set. Therefore, we split the KITTI object detection training set into training and validation/holdout sets. In the original training set, there are annotations of 28782 cars, 4487 pedestrians, and 1627 cyclists among 7481 images. As the KITTI object detection images were extracted from several videos, a splitting procedure ensured that no image from the same video was present both in the training and in the validation sets. The resulting dataset consists of 3471 images in training set and 3470 images in validation set with balanced number of annotations per class between the two sets. Within each set, however, the classes are imbalanced, as there are far more instances of the car class compared to the other two classes.

The KITTI benchmark provides point clouds collected from Velodyne LIDAR (with 64 lasers) which includes depth knowledge. In addition, calibration process can be completed prior to the fusion step because necessary information to project LIDAR points onto the 2D coordinates of images is provided. The LIDAR image obtained after the projection is initially sparse as in the upper part of Figure 3.2. To apply region proposal algorithms including both smoothing and grouping, an inpainting step is taken to provide a dense depth image.

The baseline Fast R-CNN and our CLF2 R-CNN models are trained on the training set and evaluated on the validation set using the standard AP metric: mean precision at a set of eleven equally spaced recall levels. An

object is declared as correctly detected if 50% of the box overlaps with the ground truth (70% for car). Training and testing of the model is computed on a machine with Nvidia K40. The number of iteration in training is chosen as 80000 and we trained the last two fully connected layers of popular pretrained deep CNN models with different width of networks: CaffeNet or AlexNet (Small) (Krizhevsky et al., 2012; Girshick et al., 2014) and VGG_M (Medium), a medium size of VGGNet specified in Chatfield et al. (2014). The evaluation set is pre-partitioned by KITTI into three groups, easy, moderate and hard, where difficulty is defined based on size, occlusion and truncation levels.

3.3.2 Analysis on region proposals from Selective Search

We first analyzed the ROIs proposed by Selective Search in several ways. When training the network, a ROI is marked as a positive sample if the IOU, overlap ratio with a ground truth bounding-box, is at least 0.5 (and 0.7 for the car class). The number of images that only contain ROIs with IOU less than 0.5 ($N(\#[\text{IOU}_+]{=}0)$) is counted and provided in Table 3.1 to see how many images have no additional positive ROIs. This value is also critical in testing because object detection cannot be processed properly without good ROIs; even a perfect object classifier cannot detect an object from a ROI box containing almost nothing or only a small portion of the object. Without LIDAR fusion, 27.7% of images in training set and 17.7% in validation set satisfied this condition. Then we decided to harness LIDAR in improving the ability to propose more positive ROIs. By applying Selective Search on dense

LIDAR images, 8.9% and 9.7% of images with no positive ROI sample are recovered to have positive ROIs in training and validation set respectively, thereby showing a substantial improvement.

Table 3.1: Analysis on quality of region proposals provided by Selective Search (with and without LIDAR fusion) for the KITTI object detection Train/Validation set.

Input	#[ROIs]/img	#[IOU=0]/img	N(#[IOU$_+$]=0)	AR
		Dataset: *Train*		
RGB	1090.6	672.3	962 (27.7%)	0.68
RGB$^+$	1589.0	972.9	912 (26.3%)	0.73
RGB+LIDAR	1278.1	740.6	**876 (25.2%)**	**0.73**
		Dataset: *Validation*		
RGB	1138.9	681.6	662 (19.1%)	0.69
RGB$^+$	1670.1	994.6	621 (17.9%)	0.73
RGB+LIDAR	1323.6	733.7	**598 (17.2%)**	**0.74**

In addition, the number of ROIs per image (#[ROIs]/img) and number of ROIs with no overlapping ground truth per image (#[IOU=0]/img) are calculated to investigate the characteristics and possibility of improvement in Fast R-CNN with LIDAR. First, about a thousand ROIs on average are proposed per image using Selective Search. Among those ROIs, approximately 60% do not overlap with the ground truth objects (pedestrian, cyclist, and car) which is the forth column of Table 3.1. In testing, ROIs containing no object are useless for CNN image classifier since there is nothing to classify in the region. One possible future work to address this problem will be quickly rejecting those ROIs using information from LIDAR. Both #[ROIs]/img and

#[IOU=0]/img are increased in LIDAR fusion as additional ROIs are extracted from LIDAR.

3.3.3 Results and discussion

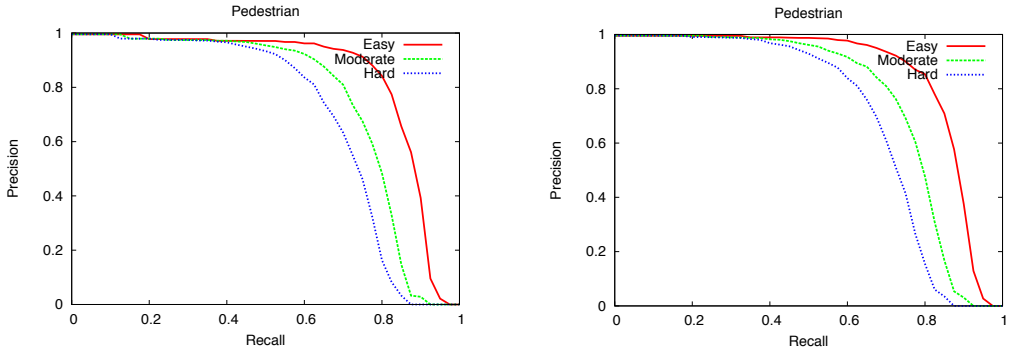


Figure 3.3: Precision-Recall curves for the KITTI pedestrian detection validation set with (*Left*) Fast R-CNN (CaffeNet, vision only) and (*Right*) CLF2 R-CNN (CaffeNet, vision + LIDAR)

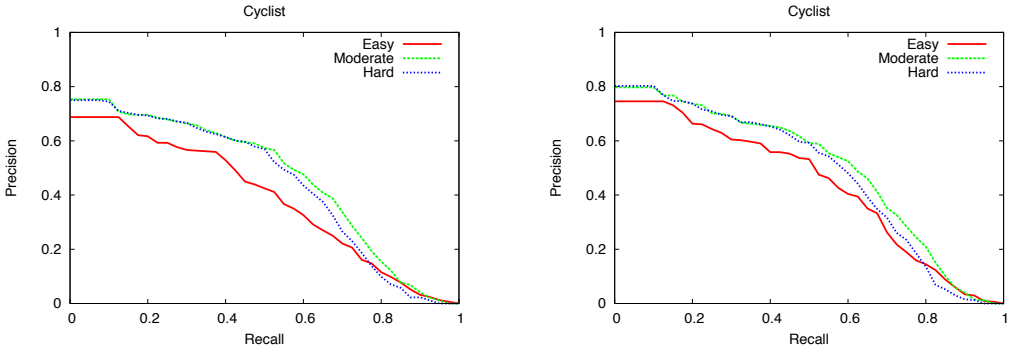


Figure 3.4: Precision-Recall curves for the KITTI cyclist detection validation set with (*Left*) Fast R-CNN (CaffeNet, vision only) and (*Right*) CLF2 R-CNN (CaffeNet, vision + LIDAR)

On the KITTI object detection validation set, we compare Fast R-CNN with our LIDAR fusion model CLF2 R-CNN. To evaluate the effectiveness of

Table 3.2: Average precision (%) for pedestrian/cyclist detection using Fast R-CNN with different settings for LIDAR support on KITTI validation set. (Test) represents a method using additional ROIs from LIDAR only in Testing and (CLF2 R-CNN) indicates our model trained with these additional ROIs.

Method	Easy	Moderate	Hard
Task: *Pedestrian Detection*			
Fast R-CNN (CaffeNet)	82.01	73.78	**67.98**
Fast R-CNN (CaffeNet) + LIDAR (Test)	81.60	72.75	66.99
CLF2 R-CNN (CaffeNet)	**82.73**	**74.12**	67.92
Fast R-CNN (VGG_M)	**84.69**	**76.50**	**71.49**
Fast R-CNN (VGG_M) + LIDAR (Test)	83.90	75.67	70.11
CLF2 R-CNN (VGG_M)	84.43	74.76	69.32
Task: *Cyclist Detection*			
Fast R-CNN (CaffeNet)	38.24	46.04	44.19
Fast R-CNN (CaffeNet) + LIDAR (Test)	37.29	44.70	43.06
CLF2 R-CNN (CaffeNet)	**42.69**	**49.08**	**47.52**
Fast R-CNN (VGG_M)	51.14	**54.03**	**52.58**
Fast R-CNN (VGG_M) + LIDAR (Test)	49.91	52.79	51.47
CLF2 R-CNN (VGG_M)	**52.34**	52.49	50.99

newly extracted ROIs from LIDAR in training, results obtained from training only with RGB and testing with additional ROIs from LIDAR are also provided in Table 3.2. Also Figure 3.3 and 3.4 show the precision-recall curves with CaffeNet on pedestrian and cyclist detection respectively, comparing Fast R-CNN and our CLF2 R-CNN.

Our model with CaffeNet outperformed Fast R-CNN with only vision data both in cyclists and pedestrians. In particular, it improved cyclist detection with 3∼4 AP (%) score and also yields a small improvement for pedestrians.

Furthermore, the positive influence of LIDAR fusion in training convolutional network can be identified from the AP score difference between two LIDAR support levels: Test vs. Train&Test. However, a model based on a wider CNN architecture, VGG_M, showed a slight decrease in the performance for the most of tasks in pedestrian and cyclist detection. But such wider designs are much more computationally demanding. Although LIDAR provides more positive ROIs to the CNN, it is possible that those ROIs have poor characteristics when viewed in RGB space. Additional ROIs found in depth image space may become a potential confusion factor for well-performing CNN architectures; overall performance with medium size convolutional network shows better results.

Table 3.3: Average precision (%) for car detection using Fast R-CNN with different LIDAR supports including CLF2 R-CNN on KITTI validation set.

Method	Easy	Moderate	Hard
Task: *Car Detection*			
Fast R-CNN (CaffeNet)	86.15	**88.01**	**79.14**
Fast R-CNN (CaffeNet) + LIDAR (Test)	91.04	87.76	78.86
CLF2 R-CNN (CaffeNet)	**92.02**	87.98	79.08
Fast R-CNN (VGG_M)	90.46	88.92	79.95
Fast R-CNN (VGG_M) + LIDAR (Test)	**93.57**	88.75	79.79
CLF2 R-CNN (VGG_M)	92.47	**88.99**	**80.01**

Compared to the car detection results in Table 3.3, pedestrian and cyclist detection models performed poorly, especially for the cyclist case. One main reason is the lack of a sufficient number of images. The car class has the most number of examples to train on, approximately 15k, while cyclist only

includes 0.8k ground truth data. Nevertheless, effect of LIDAR fusion in the region proposal stage for deep learning can be discovered from this reliable car detection task. Performances in easy tasks are increased by 2∼4 AP (%) while results for moderate and hard showed similar values among the three different methods. It indicates the supportive power of LIDAR sensor when vision algorithms miss objects that can be easily detected by humans.

Example images where the proposed fusion method gives a higher prediction probability or more accurate bounding-box are presented in Figure 3.5. One thing to notice is that both images are under low lighting conditions; robustness against illumination is a merit of LIDAR.

3.4 Conclusion

This chapter presents a new non-motorized road user detection system which incorporated both optical camera and LIDAR information to increase both accuracy and location of the objects of interest. Specifically we showed the possibility of applying deep learning technique to collision warning and collision avoidance modules. In the proposed system, our CLF2 R-CNN with a small sized CNN model CaffeNet does quite well, while being computationally less demanding than several other proposed CNN architectures. Although the overall performance is yet insufficient for deployment in onboard intelligent transportation systems, our algorithm provides another step towards the goal of designing safe and smart transportation systems of the future. It surely motivates studies for further improvements, possibly incorporating multiple

Figure 3.5: Pedestrian (*upper*) and cyclist (*lower*) detection result examples: ground truth (*red*), Fast R-CNN with CaffeNet (*blue*), and CLF2 R-CNN with CaffeNet (*lime*). Numbers indicate predicted class probabilities.

sensors and sensor types while keeping the computational requirements in check, to eventually result in a deployable working system for safer transportation systems.

Chapter 4

Object Detection with Simplified Depth Prediction

Pedestrian detection is an essential and speed-critical task for self-driving cars. Vehicle detection and prediction can possibly be aided with standard traffic rules, since a vehicle ignoring these rules would be difficult for even skilled drivers to react to, and at fault for any collision. However, pedestrians are liable to ignore traffic rules, and move slowly enough that most collisions between car and pedestrian would be considered preventable by a skilled driver. Thus pedestrian detection algorithms need to provide precise *3D position* in a short time.

It is widely accepted that self-driving vehicles will benefit from multiple types of sensors that provide complementary information. For instance, deep learning algorithms show surprisingly good object detection and identification performance with only RGB images from a single camera, but this perfor-

This chapter has been published as: Taewan Kim, Michael Motro, Patrícia Lavieri, Saharsh Samir Oza, Joydeep Ghosh, and Chandra Bhat, "Pedestrian Detection with Simplified Depth Prediction", IEEE 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, Maui, Hawaii, USA. The author of the dissertation contributed to problem formulation, data collection, model development, implementation, and empirical evaluation.

mance is contingent on lighting (as shown in the previous chapter) and other image quality factors. Image data also cannot determine the distance of a detected object, unless this is estimated after object detection using contextual knowledge.

LIDAR is robust against different illumination conditions and provides accurate distance readings. However, even commercial LIDAR of the highest resolution provide sparse information compared to RGB images, and there are fewer public datasets of the scale necessary to train deep learning algorithms for object identification - for instance, distinguishing pedestrians from stationary objects of a similar size. Resolution in LIDAR is costly, with high resolution units currently priced higher than the vehicles on which they would be installed.

In this chapter, we provide LIDAR and RGB image data directly into a deep learning network to capture complementary information in a single model. Rather than determine the full shape of each object by returning a 3D box, our algorithm predicts a 2D box in the image plus a single depth value for each pedestrian. In this way, the method provides the standard output of an image-based object detection algorithm while also providing the necessary distance information for autonomous driving. This output can be easily translated to absolute or relative 3D position for input to a tracking, prediction, or planning module. Furthermore, by only adding a single regression target, the modifications to a standard image object detection network are minimized. The amount of data needed to adequately train the modification (using a pre-trained original network) is also substantially reduced. Our implementation

of this algorithm uses a lower-resolution LIDAR sensor, specifically, the 16-layer VLP-16.

The remaining parts of this chapter are organized as follows. The collection of a novel training dataset is described in Section 4.1 with specifications. Then our deep fusion-based pedestrian detection algorithm is explained in Section 4.2. Experimental results will be covered in Section 4.3 before the conclusion in Section 4.4.

4.1 SvDPed Dataset

Detecting pedestrians in images or video is a classic problem of computer vision, with several datasets for training and comparison such as the Caltech pedestrian dataset (Dollar et al., 2012) and the MOT Challenge datasets (Milan et al., 2016). Estimating the 3-dimensional layout of an area is another classic problem, with datasets such as ScanNet (Dai et al., 2017) focusing on small indoor spaces. Datasets of 3D pedestrian position, the intersection of the two problems, are typically collected using distance-finding sensors. Most notably, the KITTI dataset (Geiger et al., 2012) includes rectangular object bounds derived manually from 64-laser LIDAR returns. Also, one of the most recent dataset nuScenes (Caesar et al., 2019) provides much larger data with various weather conditions and multimodal sensors including 32-laser LIDAR point clouds. The Daimler Pedestrian Path Prediction dataset uses stereo vision to compute distance for each pedestrian (Schneider and Gavrila, 2013).

Neither of these datasets include low-resolution LIDAR data[1], so our method was trained and evaluated with newly collected data, SvDPed (Single-valued Distance Pedestrian) dataset. Our distance ground truth values are not measured with a high-resolution on-car sensor, as positioning error from these sensors—due to natural sensor limitations or differences in sensor timing or calibration—would be included in the ground truth and positively bias the evaluation of detection algorithms based on those sensors. Additionally, it is difficult to gather ground truth for pedestrians who are substantially occluded from the view of on-car sensors. We instead use video from a nearby stationary tower to determine the absolute position of each pedestrian, and differential GNSS to determine the absolute position of the car.

4.1.1 Collection process

The position of the stationary tower is determined with a GNSS unit that is also used as a reference for the car's GNSS, so that shared atmospheric positioning error is not included in the ground truth. The real-time kinematic positioning algorithm of the RTKlib package (Takasu, 2013) is applied post-collection with the GNSS data. The orientation of the tower is determined using the car and its detected positions over a period of time. The orientation of the car is initially determined from its motion and refined using the position of the tower in the car image. Pedestrian bounding boxes can be gathered from

[1]One possible way to obtain low-resolution data is to downsample 64 lasers of KITTI. This method is applied to the evaluation in Chapter 5.

the tower's view with high precision. A high performance but slow (∼1s/image) algorithm, Faster R-CNN (Ren et al., 2015) with NASNet (Zoph et al., 2017), is used to generate reliable bounding boxes.

The bottom of each box is mapped to the ground accounting for elevation change as shown in the second step of Figure 4.1. The coordinates of each pedestrian are then projected to the car's video. Finally, pedestrians are detected in the car's video using the same Faster R-CNN model. Then each bounding box is matched to the coordinate projections by finding the optimal bipartite matching. Thus, detected pedestrians from the car's camera are assigned distance values. The projected pedestrians from the tower are not directly used as a ground truth because the height and width of a pedestrian in one camera's view are difficult to transform accurately to another's view. Instead, we use high-quality bounding boxes generated from the Faster R-CNN's detection on the cars video as ground truth. Road elevation and relative sensor positions were determined manually prior to processing the ground truth. Figure 4.1 shows the overall data collection process step by step.

Some car-detected pedestrians could not be matched to tower-detected pedestrians because they were out of view of the tower (for instance, occluded by another nearby pedestrian) or there was excessive error in the car's pose estimate. These cases are specifically annotated and avoided during training.

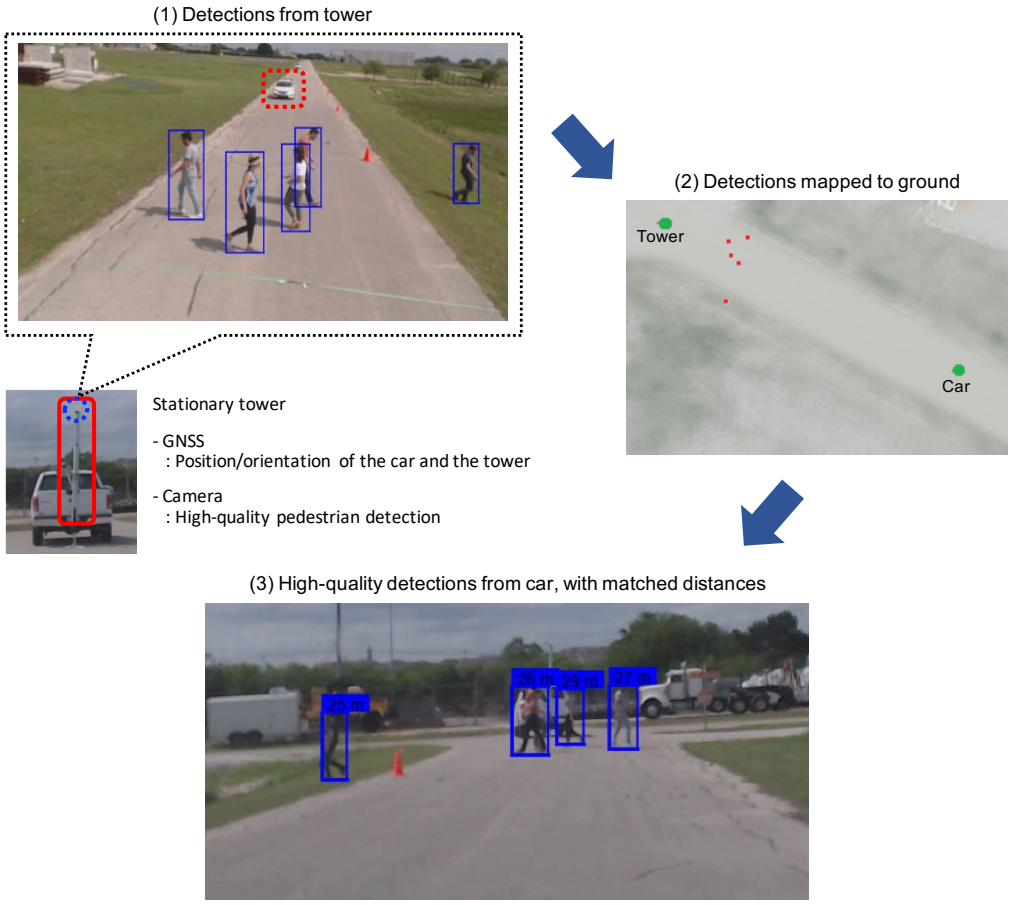


Figure 4.1: Visualizations of distance annotation pipeline

4.1.2 Calibration tool

For both generating a dataset and implementing a deep sensor-fusion model, registration or calibration of sensors has to meet certain quality. Therefore, we created a graphic system that displays a RGB camera image with LIDAR points overlaid and allows the user to adjust the sensors' positions and angles until a qualitatively good match is achieved (Figure 4.2). Few images

with several people standing in front of the vehicle, at varying distances, are sufficient to calibrate.

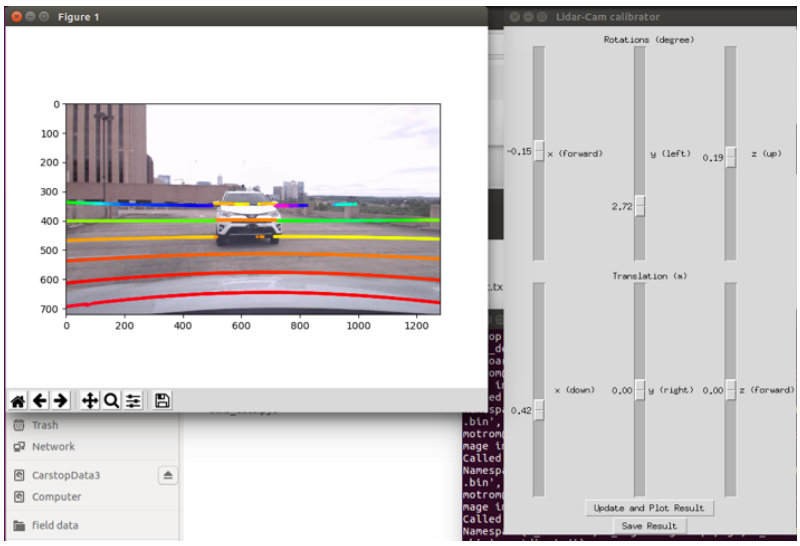


Figure 4.2: Screenshot of a developed calibration tool in GUI

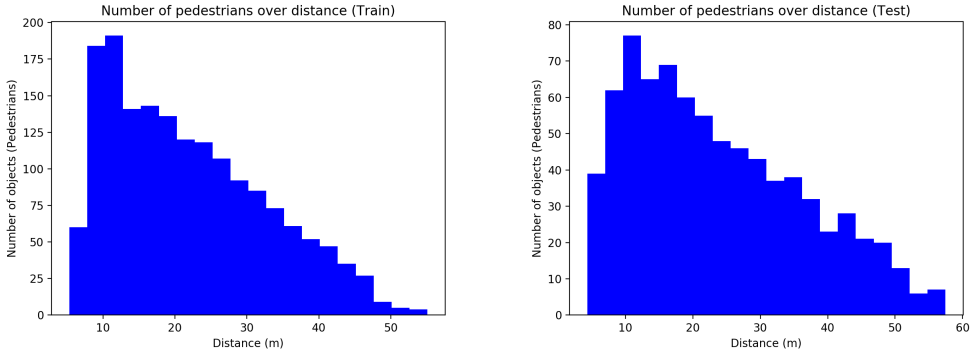
4.1.3 Resulting data

Data was collected in three locations. The first two locations were used as training data, and the third was used for evaluation. At each location, one car approached up to six pedestrians for six or seven repetitions, resulting in 20 approaches and 145 seconds of data. Five ground truth detections were made per second. Table 4.1 shows the number of images and valid targets, and Figure 4.3 depicts the distribution of collected pedestrians over distances[2].

[2]The SvDPed dataset is available at http://ideal.ece.utexas.edu/dataset/svdped.zip with intrinsic and extrinsic calibration parameters for an RGB camera and LIDAR.

Table 4.1: Specifications of the SvDPed dataset

	Train	Test
# Images	745	250
# Pedestrians	2802	915
# Pedestrians with distance	1690	789



(a) Training dataset (b) Test dataset

Figure 4.3: Histogram of number of pedestrians with ground truth distance values. *x-axis*: Distance (m). *y-axis*: Number of objects (pedestrians) in the dataset for corresponding distance ranges.

4.2 Deep Fusion Model for Pedestrian Detection

4.2.1 LIDAR preprocessing

3D LIDAR point clouds are projected onto 2D image space by using calibration matrices, so they can be easily input to conventional 2D convolutional networks. In our model, LIDAR front image is composed of two channels where pixel values are related to distance values. If a LIDAR point is represented as (x_p, y_p, z_p) where the z-axis has a forward direction, the first channel encodes

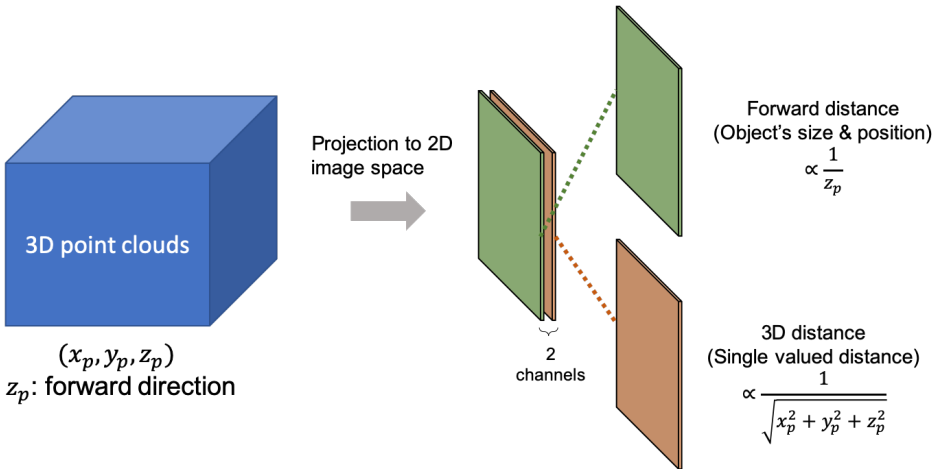


Figure 4.4: Preprocessing steps for LIDAR 3D point clouds.

the z-axis distance z_p (forward distance), and the second channel encodes $\sqrt{x_p^2 + y_p^2 + z_p^2}$, the total 3D distance.[3] These values are in a sense redundant as one can be transformed into the other given the corresponding bounding box, but z-axis distance is linearly proportional to relative object sizes and positions while total distance more directly corresponds to the prediction target. We provide both so that the network may easily utilize both. The distance value of each pixel is encoded to the range [0,255] by min-max scaling decreasing with distance. Then this LIDAR front image is preprocessed with the techniques usually applied to RGB image inputs, such as a random cropping and a random horizontal flipping. See Figure 4.4 for the two channels we use.

Our algorithm also uses another concept called initial max pooling (IMP). As LIDAR depth images are sparse, max pooling is used to upsample

[3]LIDAR depth values are clipped to the range from 2m to 75m.

sparse images, motivated by RegNet (Schneider et al., 2017) A 10×3 kernel is used as LIDAR images have more sparsity along vertical axis.

4.2.2 SSDFusion with deep learning

SSD framework generates a fixed set of boxes from multiple output layers to obtain prediction in various scales. We use batch-normalized Inception (Ioffe and Szegedy, 2015) as a baseline network to extract features. Preprocessed 2-channel LIDAR front image passes through convolution and max pooling layers, then the output features are concatenated to the ones from a RGB image. One additional 1×1 convolution layer is added to compress the channel size to meet the shape of a corresponding input layer of Inception. The rest of the architecture follows SSD with Inception except for an additional box predictor with a distance regression module. An overview of our network design is depicted in Figure 4.5.

Two parts are important in devising this model: (a) the size of output channels in LIDAR-related layers, (b) the number of convolution and layers to be passed before the merging stage. We tried two different architectures and selected channel size based on the performance. Numbers in a tuple represents following parameters: (*2D kernel*, *stride*, *output channel*).

i) SSDFusion1

   Conv(7×7, 2, 16) $\rightarrow$ MaxPool(3×3, 2, N/A)

ii) SSDFusion2

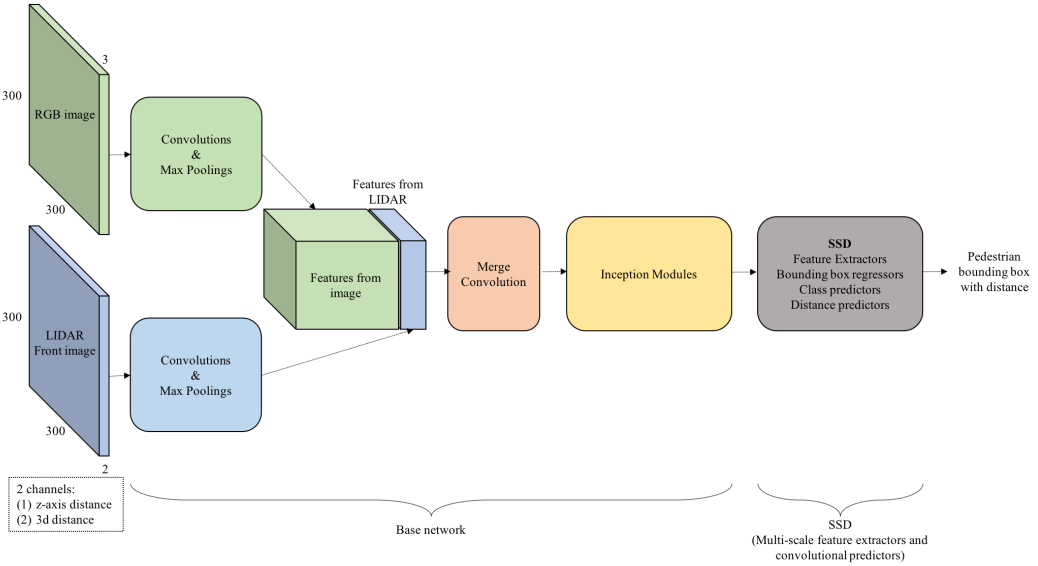


Figure 4.5: Architecture of a SSD with incorporation of LIDAR (SSDFusion).

$$\text{Conv}(7 \times 7,\ 2,\ 4) \rightarrow \text{MaxPool}(3 \times 3,\ 2,\ \text{N/A})$$
$$\rightarrow \text{Conv}(1 \times 1,\ 1,\ 8) \rightarrow \text{MaxPool}(3 \times 3,\ 2,\ \text{N/A})$$

4.2.3 Distance prediction

To train a model for distance prediction, a distance loss must be defined for each object in addition to the localization ($\mathcal{L}_{loc}$) and classification ($\mathcal{L}_{cls}$) losses. As not all objects in the training set have corresponding ground truth distances, the distance loss ($\mathcal{L}_{dist}$) is only calculated for objects with depth values. For given ground truth y and predicted objects $\hat{y}$, a total loss $\mathcal{L}$ can be defined as in (4.1). y_c represents the ground truth class, and $\hat{y}_s$ indicates per-class scores of the matched objects. y_b and $\hat{y}_b$ represents ground truth and predicted bounding boxes, respectively. Also, y_d and $\hat{y}_d$ are true and predicted

distances. An indicator function $\mathbb{I}(y_d > 0)$ controls the distance loss to be considered only if the true value exists. In practice, we use loss weights $c_{loc} = 1$ and $c_{dist} = 0.5$.

$$\mathcal{L}(y, \hat{y}) = \mathcal{L}_{cls}(y_c, \hat{y}_s) + c_{loc}\mathcal{L}_{loc}(y_b, \hat{y}_b) + \mathbb{I}(y_d > 0) \cdot c_{dist}\mathcal{L}_{dist}(y_d, \hat{y}_d) \tag{4.1}$$

4.3 Experimental Results

4.3.1 Training

Layers related to LIDAR data have to be trained from scratch. Therefore, only box predictors (convolution units for classification, bounding box regression, and distance prediction from the extracted features), layers for LIDAR, and merge layers are trained for the first 15k learning steps, while the pretrained weights are frozen. Then all weights are trained for additional 25k steps to fine-tune the network. For fair comparison we fine-tuned the SSD Inception model for 40k steps with only RGB images. Weights of the networks are initialized by using models pretrained on MS COCO dataset (Lin et al., 2014) provided by TensorFlow Object Detection API[4]. For extensive comparison, performance of the model pretrained only on MS COCO dataset (SSD Inception Plain), and fine-tuned on our SvDPed dataset (SSD Inception) are also reported. RMSprop optimizer is used with the following piecewise learning rates:

[4] https://github.com/tensorflow/models/tree/master/research/object_detection

i) SSD (RGB only): .004 (step 0), 0.0001 (step 20k)

ii) (IMP) SSDFusion1 & 2

(a) Partial learning: .004 (step 0), 0.0001 (step 10k)

(b) Fine-tuning: .004 (step 15k), 0.0001 (step 30k)

4.3.2 Distance prediction from LIDAR

To assess the quality of our algorithm's depth prediction, we implemented two approaches as baselines to extract a single distance value from the LIDAR points in a bounding box: (a) LIDAR-ClosestPoint and (b) LIDAR-Clustering. Ground truth bounding boxes are provided as input to the models using both approaches to solely focus on the effectiveness of using LIDAR points in predicting distance. First, LIDAR-ClosestPoint selects a minimum 3D distance value among the LIDAR points inside the bounding box as a prediction. Another method LIDAR-Clustering uses a clustering algorithm on the z-axis distances of the points included in the box. Then a cluster with the minimum average distance is selected to predict the distance by averaging the 3D distances of the points in the cluster[5]. Clustering can fail if the number of points in the box is too small. In this case, we predicted the distance as the closest one.

For both approaches, depth prediction is set to 20m, roughly the mean

[5]We also tried to select a cluster with the maximum number of elements, but the RMSE values were about 2m higher.

distance in the dataset, whenever there is no point inside the box. Results are presented in Table 4.2 and 4.3. Note that the reported results are evaluated using all ground truth bounding boxes, whereas RMSE evaluation on our fusion algorithms are only considering true positive detections.

4.3.3 Results

Average precision (AP) is used as an evaluation metric for object detection, where a detection is declared as a true positive if the detection box and the ground truth box overlaps with IOU greater than or equal to 0.5 (AP@0.5IOU). For distance prediction, we use root mean square error between the distances of the true positive boxes and corresponding ground truths (RMSE@0.5IOU), only when the ground truth value is available ($y_d > 0$).

Table 4.2 and 4.3 show our experimental results including different baseline models. SSDFusion1 Inception model shows the best performance in test data. AP 51.88 is achieved with a smaller gap between train and test scores than the scores of SSD Inception fine-tuned on our dataset. Also, we believe that RMSE 3.52 can be improved by training with more data. Although baseline methods for distance prediction works well with around 2∼3m error, they rely heavily on the quality of the bounding boxes since they only use LIDAR points within these boxes. The baselines were given the true bounding boxes, and so their performance is overestimated to some degree. On the other hand, our fusion algorithm jointly predicts distance and detect object. SSDFusion1 shows descent distance prediction results while almost preserving

the performance of object detection. This may come from the simplicity of single valued distance prediction.

Table 4.2: AP of object detection and root mean square error (RMSE) of distance prediction results for the detected boxes with IOU$\geq$ 0.5 on *Training* dataset. Detected bounding boxes with confidence score less than 0.3 are disregarded in evaluation.

Method	AP@0.5IOU (%)	RMSE@0.5IOU (m)
Dataset: *Train*		
(Object Detection Only)		
SSD Inception Plain	42.74	N/A
SSD Inception	**82.07**	N/A
(Distance Prediction Only)		
LIDAR-ClosestPoint	N/A	**3.31**
LIDAR-Clustering	N/A	6.06
(Joint Prediction)		
SSDFusion1 Inception	79.11	0.72
SSDFusion2 Inception	**79.42**	**0.71**
IMP SSDFusion1 Inception	71.18	4.50
IMP SSDFusion2 Inception	71.75	4.43

The overall performance of each model is not well generalized in the test set. It is likely that the size of our dataset leading to overfitting. Additional training data should alleviate this issue. Figure 4.6 shows some examples of detection with depth prediction. Compared to the ground truth boxes, our SSDFusion algorithm has more difficulty detecting pedestrians at far distances, which is also happening in SSD Inception model. Nevertheless, if an object is detected, our model will still predict the distance even if the number of

Table 4.3: AP of object detection and root mean square error (RMSE) of distance prediction results for the detected boxes with IOU$\geq$ 0.5 on *Test* dataset. Detected bounding boxes with confidence score less than 0.3 are disregarded in evaluation.

Method	AP@0.5IOU (%)	RMSE@0.5IOU (m)
	Dataset: *Test*	
(Object Detection Only)		
SSD Inception Plain	32.80	N/A
SSD Inception	**51.54**	N/A
(Distance Prediction Only)		
LIDAR-ClosestPoint	N/A	**1.82**
LIDAR-Clustering	N/A	1.98
(Joint Prediction)		
SSDFusion1 Inception	**51.88**	**3.52**
SSDFusion2 Inception	47.08	3.82
IMP SSDFusion1 Inception	49.38	6.44
IMP SSDFusion2 Inception	46.05	6.30

LIDAR points sensed by the LIDAR is too small. Preprocessing LIDAR front image at the beginning with max pooling does not contribute to the model's ability despite its computational cost. As low-resolution LIDAR is used in our system, a depth map cannot provide rich details and it might be better to let convolutional layers deal with sparse inputs. The inference speed of our model is around 40fps on a GTX 1070 GPU, and the rate of VLP-16 was fixed to 10Hz in data collection.[6]

[6]Code is available at https://github.com/twankim/svdped

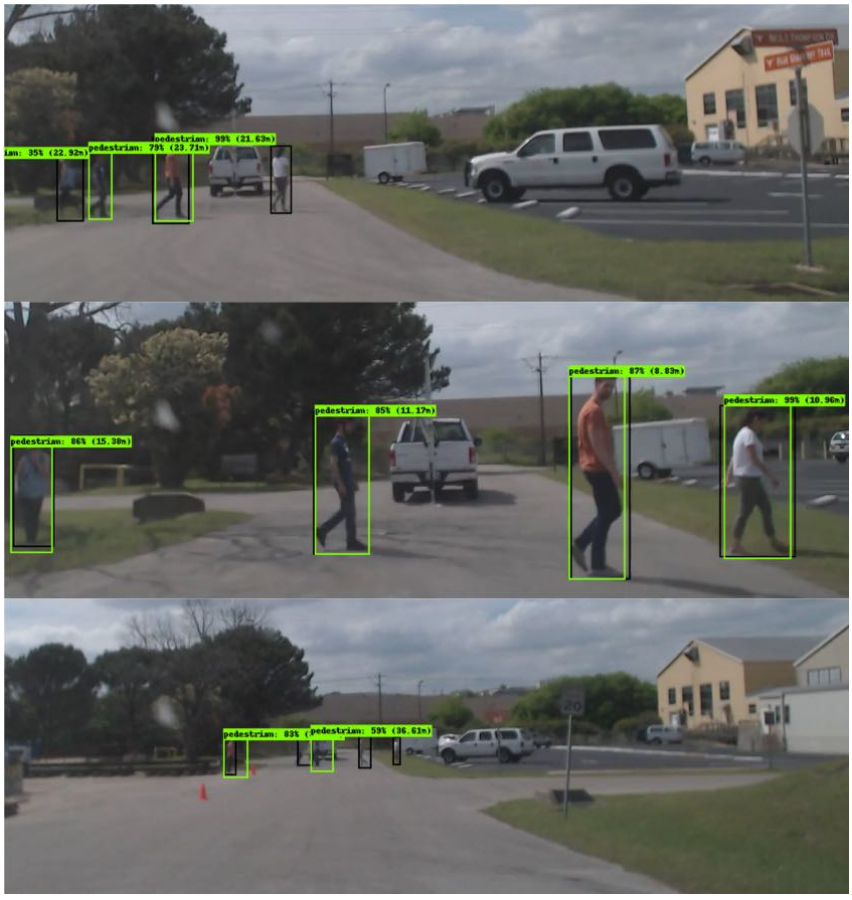


Figure 4.6: Pedestrian detection results with distance prediction. *Black boxes*: ground truth, *Green boxes*: Detection from SSDFusion Inception.

4.4 Conclusion

This chapter presents a new deep learning-based pedestrian detection algorithm, using low-resolution LIDAR and RGB video. The primary innovation is to merge the detection of pedestrians in images and the prediction of their distance from the vehicle. Existing object detection algorithms can be easily extended as the task of predicting single distance values is more efficient

than predicting exact 3D bounding boxes. Our algorithm is trained on newly collected data, which has ground truth depth labels obtained from GNSS and camera poses rather than from high-resolution on-car sensors. The proposed model can be improved by providing more training data of faraway pedestrians. Our collection method is capable of gathering accurate data regardless of distance, and together with the dataset release, is part of the contribution of this work.

Chapter 5

Robustness Against Single Source Faults

Two benefits are expected when fusion-based learning models are selected for a given problem. First, given adequate data, more information from multiple sources can enrich the model's feature space to achieve higher prediction performance, especially, when different input sources provide *complementary information* to the model. This expectation coincides with a simple information theoretic fact: if we have multiple input sources $X_1, \cdots, X_m$ and a target variable Y, mutual information $I(;)$ obeys $I(Y; X_1, \cdots, X_m) \geq I(Y; X_i) \ \ (\forall i \in [m])$.

The second expected advantage is increased robustness against single source faults, which is the primary concern of our work. An underlying intuition comes from the fact that different sources may have *shared information* so one sensor can partially compensate for others. This type of robustness is critical in real-world fusion models, because each source may be exposed to different

This chapter has been published as: Taewan Kim, and Joydeep Ghosh, "On Single Source Robustness in Deep Fusion Models", arXiv preprint arXiv:1906.04691, 2019. The author of the dissertation contributed to problem formulation, analyses, model development, implementation, and empirical evaluation.

types of corruption but not at the same time. For example, LIDAR used in autonomous vehicles work fine at night whereas RGB cameras do not. Also, each source used in the model may have its own sensing device, and hence not necessarily be corrupted by some physical attack simultaneously with others. It would be ideal if the structure of machine learning based fusion models and shared information could compensate for the corruption and automatically guarantee robustness without additional steps.

This chapter shows that a fusion model needs a supplementary strategy and a specialized structure to avoid vulnerability to noise or corruption on a single source. Our contributions are as follows:

- We show that a fusion model learned with a standard robustness is not guaranteed to provide robustness against noise on a single source. Inspired by the analysis, a novel loss is proposed to achieve the desired robustness (Section 5.2).
- Two efficient training algorithms for minimizing our loss in deep fusion models are devised to ensure robustness without impacting performance on clean data (Section 5.3.1).
- We introduce a simple but an effective fusion layer which naturally reduces error by applying ensembling to latent convolutional features (Section 5.3.2).

We apply our loss and the fusion layer to a complex deep fusion-based 3D object detector used in autonomous driving for further investigation in practice.

Note that our findings can be easily generalized to other applications exhibiting intermittent defects in a subset of input sources.

5.1 Related Work

Deep fusion models have been actively studied in object detection for autonomous vehicles. There exist two major streams classified according to their algorithmic structures: two-stage detectors with R-CNN (Region-based Convolutional Neural Networks) technique (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Dai et al., 2016; He et al., 2017), and single stage detectors for faster inference speed (Redmon et al., 2016; Redmon and Farhadi, 2017; Liu et al., 2016).

Earlier deep fusion models extended Fast R-CNN (Girshick, 2015) to provide better quality of region proposals from multiple sources including our CLF2 R-CNN (Chapter 3) (Kim and Ghosh, 2016; Braun et al., 2016). With a high-resolution LIDAR, point cloud was used as a major source of the region proposal stage before the fusion step (Du et al., 2017), whereas F-PointNet (Qi et al., 2018) used it for validating 2D proposals from RGB images and predicting 3D shape and location within the visual frustum. MV3D (Chen et al., 2017) extended the idea of region proposal network (RPN) (Ren et al., 2015) by generating proposals from RGB image, and LIDAR's front view and BEV (bird's eye view) maps. Recent works tried to remove region proposal stages for faster inference and directly fused LIDAR's front view depth image (Kim et al., 2018b) or BEV image (Wang et al., 2018) with RGB images. ContFuse

(Liang et al., 2018) utilizes both RGB and LIDAR's BEV images with a new continuous fusion scheme, which is further improved in MMF (Liang et al., 2019) by handling multiple tasks at once. Our experimental results are based on AVOD (Ku et al., 2018), a recent open-sourced 3D object detector that generates region proposals from RPN using RGB and LIDAR's BEV images.

Compared to the active efforts in accomplishing higher performance on clean data, very few works have focused on robust learning methods in multi-source settings to the best of our knowledge. Adaptive fusion methods using gating networks weight the importance of each source automatically (Mees et al., 2016; Valada et al., 2017), but these works lack in-depth studies of the robustness against single source faults. A recent work proposed a gated fusion at the feature level and applied data augmentation techniques with randomly chosen corruption methods (Kim et al., 2018a). In contrast, our training algorithms are surrogate minimization schemes for the proposed loss function, which is grounded from the analyses on underlying weakness of fusion methods. Also the fusion layer proposed in this chapter focuses more on how to mix convolutional feature maps *channel-wise* with simple trainable procedures.

5.2 Single Source Robustness of Fusion Models

5.2.1 Regression on linear fusion data

To show the vulnerability of naive fusion models, we introduce a simple data model and a fusion algorithm. Suppose y is a linear function consisting of three different inherent (latent) components $z_i \in \mathbb{R}^{d_i}$ $(i \in \{1, 2, 3\})$. There are

two input sources, x_1 and x_2. Here, ψ's are unknown functions.

$$y = \sum_{i=1}^{3} {\beta_i}^T z_i,$$
$$\text{where} \;\; z_1 = \psi_1(x_1), \;\; z_2 = \psi_2(x_2), \;\; z_3 = \psi_{3,1}(x_1) = \psi_{3,2}(x_2) \tag{5.1}$$

Our simple data model simulates a target variable y relevant to two different sources, where each source has its own special information z_1 and z_2 and a shared one z_3. For example, if two sources are obtained from an RGB camera and a LIDAR sensor, one can imagine that any features related to objectness are captured in z_3 whereas colors and depth information may be located in z_1 and z_2, respectively. Our objective is to build a regression model by effectively incorporating information from the sources (x_1, x_2) to predict the target variable y.

Now, consider a fairly simple setting $x_1 = [z_1; z_3] \in \mathbb{R}^{d_1+d_3}$ and $x_2 = [z_2; z_3] \in \mathbb{R}^{d_2+d_3}$, where $(\psi_1, \psi_2, \psi_{3,1}, \psi_{3,2})$ can be defined accordingly to satisfy (5.2.1). A straightforward fusion approach is to stack the sources, i.e. $x = [x_1; x_2] \in \mathbb{R}^{d_1+d_2+2d_3}$, and learn a linear model. Then, it is easy to show that there exists a feasible *error-free model* for noise-free data:

$$\begin{aligned} f_{\text{direct}}(x_1, x_2) &= h_1^T x_1 + h_2^T x_2 \\ &= (\beta_1^T z_1 + g_1^T z_3) + (\beta_2^T z_2 + g_2^T z_3), \quad s.t.\ g_1 + g_2 = \beta_3 \end{aligned} \tag{5.2}$$

where $h_1 = [\beta_1; g_1], h_2 = [\beta_2; g_2]$. Parameter vectors responsible for the shared information z_3 are denoted by g_1 and g_2. In practice, $Y = [X_1, X_2] \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$ has to be solved for $X_1 \in \mathbb{R}^{n \times (d_1+d_3)}, X_2 \in \mathbb{R}^{n \times (d_2+d_3)}$ and $Y \in \mathbb{R}^n$ with enough

number of n data samples. Then a standard least squares solution using a pseudo-inverse gives $h_1 = [\beta_1; \beta_3/2],\ h_2 = [\beta_2; \beta_3/2]$. This is equivalent to the solution robust against random noise added to all the sources at once, which is vulnerable to single source faults (Section 5.2.2).

Unbalanced robustness (Motivation)

Suppose the true parameters of data are scalar values, i.e. $\beta_i = c_i \in \mathbb{R}$ and influence of the complementary information is relatively small, $c_1 \approx c_2$ and $c_3 \gg c_1$. Assume that the obtained error-free solution's parameters for z_3 are unbalanced, i.e. $g_1 = \Delta$ and $g_2 = c_3 - \Delta$ with some weight parameter $\Delta \ll c_3$, so that g_1 gives a negligible contribution. Then add single source corruption $\delta_1 = [\epsilon_1; \epsilon_3]$ and $\delta_2 = [\epsilon_2; \epsilon_3]$ and compute absolute difference between the true data y and the prediction from the corrupted data:

$$|y - f_{\text{direct}}(x_1 + \delta_1, x_2)| = |c_1\epsilon_1 + \Delta\epsilon_3|$$
$$|y - f_{\text{direct}}(x_1, x_2 + \delta_2)| = |c_2\epsilon_2 + (c_3 - \Delta)\epsilon_3|$$

In this case, adding noise to the source x_2 will give significant corruption to the prediction while x_1 is relatively robust because $|(c_3 - \Delta)\epsilon_3| \gg |\Delta\epsilon_3|$ for any noise ϵ_3 affecting z_3. This simple example illustrates that additional training strategies or components are indispensable to achieve robust fusion model working even if one of the sources is disturbed. The next section introduces a novel loss for a balanced robustness against a fault in a single source.

5.2.2 Robust learning for single source noise

Fusion methods are not guaranteed to provide robustness against faults in a single source without additional supervision. Also, we demonstrate that naive regularization or robust learning methods are not sufficient for the robustness later in this section. Therefore, a supplementary constraint or strategy needs to be considered in training which can correctly guide learning parameters for the desired robustness.

One essential requirement of fusion models is showing *balanced performance* regardless of corruption added to any source. If the model is significantly vulnerable to corruption in one source, this model becomes untrustworthy and we need to balance the degradation levels of different input sources' faults. For example, suppose there is a model robust against noise in RGB channels but shows huge degradation in performance for any fault of LIDAR. Then the overall system should be considered untrustworthy, because there exist certain corruption or environments which can consistently fool the model. Our loss, MaxSSN (Maximum Single Source Noise), for such robustness is introduced to handle this issue and further analyses are provided under the linear fusion data model explained in Section 5.2.1. This loss makes the model focus more on corruption of a single source, SSN, rather than focusing on noise added to all the sources at once, ASN.

Definition 5.1. *For multiple sources $x_1, \cdots, x_{n_s}$ and a target variable y, denote a predefined loss function by $\mathcal{L}$. If each source x_i is perturbed with some additive*

noise ϵ_i *for* $i \in [n_s]$, MaxSSN *loss for a model* f *is defined as follows:*

$$\mathcal{L}_{\text{MaxSSN}}(f, \epsilon) \triangleq \max_i \{\mathcal{L}\left(y, f(x_1, \cdots, x_{i-1}, x_i + \epsilon_i, x_{i+1}, \cdots, x_{n_s})\right)\}_{i=1}^{n_s}$$

Another key principle in our robust training is to *retain the model's performance on clean data*. Although techniques like data augmentation help improving a model's generalization error in general, learning a model robust against certain types perturbation including adversarial attacks may harm the model's accuracy on non-corrupt data (Tsipras et al., 2019). Deterioration in the model's ability on normal data is an unwanted side effect, and hence our approach aims to avoid this.

Random noise

To investigate the importance of our MaxSSN loss, we revisit the linear fusion data model with the optimal direct fusion model f_{direct} of the regression problem introduced in Section 5.2.1. Suppose the objective is to find a model with robustness against single source noises, while preserving error-free performance, i.e., unchanged loss under clean data. For the noise model, consider $\epsilon = [\delta_1; \delta_2]$ where $\delta_1 = [\epsilon_1; \epsilon_3]$ and $\delta_2 = [\epsilon_2; \epsilon_4]$, which satisfy $\mathbb{E}[\epsilon_i] = 0$, $Var(\epsilon_i) = \sigma^2 I$, and $\mathbb{E}[\epsilon_i \epsilon_j^T] = 0$ for $i \neq j$. Note that noises added to the shared information, ϵ_3 and ϵ_4, are not identical, which resembles direct perturbation to the input sources in practice. For example, noise directly affecting a camera lens does not need to perturb other sources.

Optimal fusion model for MaxSSN

The robust linear fusion model $f(x_1, x_2) = (w_1^T z_1 + g_1^T z_3) + (w_2^T z_1 + g_2^T z_3)$ is found by minimizing $\mathcal{L}_{\text{MaxSSN}}(f, \epsilon)$ over parameters w_1, w_2, g_1 and g_2. As shown in the previous section, any f_{direct} satisfying $w_1 = \beta_1, w_2 = \beta_2$ and $g_1 + g_2 = \beta_3$ should achieve zero-error. Therefore, overall optimization problem can be reduced to the following one:

$$\min_{g_1, g_2} \max \left\{ \mathcal{L}\left(y, f_{\text{direct}}(x_1 + \delta_1, x_2)\right), \mathcal{L}\left(y, f_{\text{direct}}(x_1, x_2 + \delta_2)\right) \right\}$$
$$s.t. \;\; g_1 + g_2 = \beta_3 \tag{5.3}$$

If we use a standard expected squared loss $\mathcal{L}(y, f(x_1, x_2)) = \mathbb{E}[(y - f(x_1, x_2))^2]$ and solve the optimization problem, the following solution $\mathcal{L}^*_{\text{MaxSSN}}$ with corresponding parameters g_1^*, g_2^* can be obtained, and there exist three cases based on the relative sizes of $||\beta_i||_2$'s.

$$(\mathcal{L}^*_{\text{MaxSSN}},\ g_1^*,\ g_2^*) = \begin{cases} (\sigma^2 ||\beta_2||_2^2,\ \beta_3,\ 0) & \text{if } ||\beta_1||_2^2 + ||\beta_3||_2^2 \leq ||\beta_2||_2^2 \\ (\sigma^2 ||\beta_1||_2^2,\ 0,\ \beta_3) & \text{if } ||\beta_2||_2^2 + ||\beta_3||_2^2 \leq ||\beta_1||_2^2 \\ \left(\sigma^2 \left(\frac{||\beta_1||_2^2 + ||\beta_2||_2^2}{2} + \frac{||\beta_3||_2^2}{4} + \frac{(||\beta_2||_2^2 - ||\beta_1||_2^2)^2}{4||\beta_3||_2^2}\right), \right. & \\ \left. \frac{1}{2}\left(1 + \frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right),\ \frac{1}{2}\left(1 - \frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right)\right) & \text{otherwise} \end{cases} \tag{5.4}$$

The three cases reflect the relative influence of each weight vector for z_i. For instance, if z_2 has larger importance compared to the rest in generating y, the optimal way of balancing the effect of noise over z_3 is to remove all the influence of z_2 in x_2 by setting $g_2 = 0$. When neither of z_1 nor z_2 dominates the importance, i.e. $\left|\frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right| < 1$, the optimal solution tries to make $\mathcal{L}\left(y, f_{\text{direct}}(x_1 + \delta_1, x_2)\right) = \mathcal{L}\left(y, f_{\text{direct}}(x_1, x_2 + \delta_2)\right)$.

Comparison with the standard robust fusion model

Minimizing loss with noise added to a model's input is a standard process in robust learning. The same strategy can be applied to learn fusion models by considering all sources as a single combined source, then add noise to all the sources at once. However, this simple strategy cannot achieve low error in terms of the single source robustness. The optimal solution to $\min_{g_1,g_2} \mathbb{E}[(y - f_{\text{direct}}(x_1 + \delta_1, x_2 + \delta_2))^2]$, a least squares solution, is achieved when $g_1 = g_2 = \frac{\beta_3}{2}$. The corresponding MaxSSN loss can be evaluated as $\mathcal{L}'_{\text{MaxSSN}} = \sigma^2 \max\left\{||\beta_1||_2^2 + \frac{1}{4}||\beta_3||_2^2, ||\beta_2||_2^2 + \frac{1}{4}||\beta_3||_2^2\right\}$. A nontrivial gap exists between $\mathcal{L}_{\text{MaxSSN}}$ and $\mathcal{L}'_{\text{MaxSSN}}$, which is directly proportional to the data model's inherent characteristics:

$$\mathcal{L}'_{\text{MaxSSN}} - \mathcal{L}^*_{\text{MaxSSN}} \geq \begin{cases} \frac{1}{4}||\beta_3||_2^2 & \text{if } \left|\frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right| \geq 1 \\ \frac{1}{4}\left|||\beta_2||_2^2 - ||\beta_1||_2^2\right| & \text{otherwise} \end{cases} \tag{5.5}$$

If either z_1 or z_2 has more influence on the target value y than the other components, single source robustness of the model trained by MaxSSN loss is better than the fusion model for the general noise robustness with an amount proportional to the influence of shared feature z_3. Otherwise, the gap's lower bound is proportional to the difference in complementary information, $|||\beta_2||_2^2 - ||\beta_1||_2^2|/4$.

Remark 5.1. *In linear systems such as the one studied above, having redundant information in the feature space is similar to multicollinearity in statistics. In this case, feature selection methods usually try to remove such redundancy.*

However, this redundant or shared information helps preventing degradation of the fusion model when a subset of the input sources are corrupted.

Remark 5.2. *Similar analyses and a loss definition against adversarial attacks (Goodfellow et al., 2015) are provided in appendix A.1.*

5.2.3 Proofs and analyses

Optimal solution for minimizing MaxSSNloss

Proof. The original $\mathcal{L}_{\text{MaxSSN}}$ loss minimization problem with an additional constraint of preserving loss under clean data can be transformed to the problem stated in (5.2.2) due to the flexibility of g_1 and g_2 under the constraint $g_1 + g_2 = \beta_3$:

$$\min_{g_1,g_2} \max \left\{ \mathcal{L}\left(y, f_{\text{direct}}(x_1 + \delta_1, x_2)\right), \mathcal{L}\left(y, f_{\text{direct}}(x_1, x_2 + \delta_2)\right) \right\}$$
$$hspace.5ems.t. \quad g_1 + g_2 = \beta_3$$

Under the expected squared loss with f_{direct} function, the loss can be evaluated,

$$\begin{aligned}
&\mathcal{L}\left(y, f_{\text{direct}}(x_1 + \delta_1, x_2)\right) \\
&= \mathbb{E}\left[\left(y - (\beta_1^T(z_1 + \epsilon_1) + g_1^T(z_3 + \epsilon_3) + \beta_2^T z_2 + g_2^T z_3)\right)^2\right] \\
&= \mathbb{E}\left[\left(\beta_1^T \epsilon_1 + g_1^T \epsilon_3\right)^2\right] \qquad \left(\because y = \sum_{i=1}^{3} {\beta_i}^T z_i\right) \\
&= \sigma^2(||\beta_1||_2^2 + ||g_1||_2^2) \qquad (\because \text{Statistical assumption on } \epsilon_i.)
\end{aligned}$$

Hence the equivalent problem (5.6) is achieved.

$$\sigma^2 \min_{g_1,g_2} \max \left\{ ||\beta_1||_2^2 + ||g_1||_2^2, ||\beta_2||_2^2 + ||g_2||_2^2 \right\} \quad s.t. \quad g_1 + g_2 = \beta_3 \tag{5.6}$$

For simple notation, substitute variables as $g = g_1, v = \beta_3, c_1 = ||\beta_1||_2^2, c_2 = ||\beta_2||_2^2$, and solve the following convex optimization problem.

$$\min_{g} \max\{||g||_2^2 + c_1, ||g - v||_2^2 + c_2\}$$

This problem can be solved by introducing a variable γ for the upper bound of the inner maximum value:

$$\min_{g,\gamma} \gamma \quad s.t. \;\; c_1 + ||g||_2^2 - \gamma \leq 0, \;\; c_2 + ||g - v||_2^2 - \gamma \leq 0$$

KKT condition gives:

$$\begin{aligned}
&\text{(Primal feasibility)} && c_1 + ||g||_2^2 - \gamma \leq 0, \;\; c_2 + ||g - v||_2^2 - \gamma \leq 0 \\
&\text{(Dual feasibility)} && \lambda_1 \geq 0, \;\; \lambda_2 \geq 0 \\
&\text{(Complementary slackness)} && \lambda_1(c_1 + ||g||_2^2 - \gamma) = 0, \\
& && \lambda_2(c_2 + ||g - v||_2^2 - \gamma) = 0 \\
&\text{(Stationary)} && \lambda_1 + \lambda_2 = 1, \;\; g = \frac{\lambda_2}{\lambda_1 + \lambda_2} v
\end{aligned}$$

Considering $\lambda_1 + \lambda_2 = 1$ and $\lambda_1, \lambda_2 \geq 0$, we first need to analyze the case $\lambda_1 = 0$. This gives $g = v$ and the complementary slackness condition to find $\gamma = c_2 + ||g - v||_2^2 = c_2$. $\lambda_2 = 0$ can be analyzed with similar steps. If both λ_1 and λ_2 are positive, the complementary slackness condition gives $\gamma = c_1 + ||g||_2^2 = c_2 + ||g - v||_2^2$, which ensures the balance of the original problem's maximum value $\max\{c_1 + ||g||_2^2, c_2 + ||g - v||_2^2\}$. This case gives $\gamma = \frac{c_1 + c_2}{2} + \frac{||v||_2^2}{4} + \frac{(c_2 - c_1)^2}{4||v||_2^2}$ with $g = \frac{1}{2}\left(1 + \frac{c_2 - c_1}{||v||_2^2}\right) v$. Therefore, we can have the result (5.4) which provides the fusion model robust against single source corruptions from random noise. □

Solution for the standard robustness loss and comparison

If random noise are added to x_1 and x_2 simultaneously, the objective of the problem becomes $\min_{g_1,g_2} \mathbb{E}[(y - f_{\text{direct}}(x_1 + \delta_1, x_2 + \delta_2))^2]$ instead of considering the MaxSSN loss. This is equivalent to minimizing $\sigma^2(||\beta_1||_2^2 + ||\beta_2||_2^2 + ||g_1||_2^2 + ||g_2||_2^2)$ subject to $g_1 + g_2 = \beta_3$, and the solution can be directly found as it is a simple convex problem, which is $g_1 = g_2 = \frac{\beta_3}{2}$. If we denote this model as f'_{direct}, then MaxSSN loss is:

$$\mathcal{L}_{\text{MaxSSN}}(f'_{\text{direct}}, \epsilon) = \mathcal{L}'_{\text{MaxSSN}} = \sigma^2 \max\left\{||\beta_1||_2^2 + \frac{1}{4}||\beta_3||_2^2, ||\beta_2||_2^2 + \frac{1}{4}||\beta_3||_2^2\right\}$$

Now, let's compute the difference $\mathcal{L}'_{\text{MaxSSN}} - \mathcal{L}^*_{\text{MaxSSN}}$.

Proof. As both term includes σ^2, let's assume $\sigma^2 = 1$ for ease of notation. Among the three cases in (5.4), consider the first case $||\beta_1||_2^2 + ||\beta_3||_2^2 \leq ||\beta_2||_2^2$.

$$\begin{aligned}
&\mathcal{L}'_{\text{MaxSSN}} - \mathcal{L}^*_{\text{MaxSSN}} \\
&\qquad = ||\beta_2||_2^2 + \frac{1}{4}||\beta_3||_2^2 - ||\beta_2||_2^2 = \frac{1}{4}||\beta_3||_2^2 \quad (\because ||\beta_2||_2^2 \geq ||\beta_1||_2^2)
\end{aligned}$$

The second case can be shown similarly. Now assume that $\left|\frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right| < 1$ holds, and let $||\beta_2||_2^2 \geq ||\beta_1||_2^2$ without loss of generality. Then we can show,

$$\begin{aligned}
&\mathcal{L}'_{\text{MaxSSN}} - \mathcal{L}^*_{\text{MaxSSN}} \\
&= ||\beta_2||_2^2 + \frac{1}{4}||\beta_3||_2^2 - \left(\frac{||\beta_1||_2^2 + ||\beta_2||_2^2}{2} + \frac{||\beta_3||_2^2}{4} + \frac{(||\beta_2||_2^2 - ||\beta_1||_2^2)^2}{4||\beta_3||_2^2}\right) \\
&= \frac{1}{2}(||\beta_2||_2^2 - ||\beta_1||_2^2)\left(1 - \frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{2||\beta_3||_2^2}\right) \\
&\geq \frac{1}{4}(||\beta_2||_2^2 - ||\beta_1||_2^2) \quad \left(\because ||\beta_2||_2^2 \geq ||\beta_1||_2^2 \text{ and } \left|\frac{||\beta_2||_2^2 - ||\beta_1||_2^2}{||\beta_3||_2^2}\right| < 1\right)
\end{aligned}$$

□

Therefore we can conclude that simply optimizing under noise added to all the input sources at the same time cannot do better than minimizing MaxSSN loss with some nonnegative gap in our linear fusion model.

5.3 Robust Deep Fusion Models

In simple linear settings, our analyses illustrate that using MaxSSN loss can effectively minimize the degradation of a fusion model's performance. This suggests a training strategy for complex deep fusion models to be equipped with robustness against single source faults. A principal factor considered in designing a common framework for our algorithms is the preservation of model's performance on clean data while minimizing a loss for defending corruption. Therefore, our training algorithms use *data augmentation* to encounter both clean and corrupted data. The second way of achieving robustness is to take advantage of the fusion method's structure. A simple but effective method of mixing convolutional features coming from different input sources is introduced later in this section.

5.3.1 Robust training algorithms for single source noise

In the previous section, we solve problem (5.2.2) by optimizing over flexible parameters g_1 and g_2. If the parts of input sources contributing to z_3 are known, then indeed we can achieve this goal. In practice however, it is difficult to know which parts of an input source (or latent representation) are related to shared information and which parameters are flexible. Therefore, our

common training framework alternately provides *clean samples* and *corrupted samples* per iteration to preserve the original performance of the model on uncontaminated data. We also try *fine-tuning* only a subset of the model's parameters, $\boldsymbol{\theta}_{\text{fusion}} \subset f$, to preserve essential parts for extracting features from normal data. Although this strategy is similar to optimizing over only g_1 and g_2 in our linear fusion case, training the whole network from the beginning shows better performance in practice. See Appendix A.2 for a detailed comparison.

On top of this strategy, one standard robust training scheme and two algorithms for minimizing MaxSSN loss are introduced for handling robustness against noise in different sources.

Standard robust training method

Algorithm 1 TrainASN

for $i_{\text{iter}} = 1$ to m **do**
  Sample $(y, \{x_i\}_{i=1}^{n_s})$
  if $i_{\text{iter}} \equiv 1 \pmod 2$ **then**
    Generate noise $\epsilon_i = \varphi_i(x_i),\ \forall i \in [n_s]$
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_i + \epsilon_i\}_{i=1}^{n_s}))$
  else
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_i\}_{i=1}^{n_s}))$
  end if
  Update f using $\nabla \mathcal{L}^{(i_{\text{iter}})}$
end for

A standard robust training algorithm can be developed by considering all n_s sources as a single combined source. Given noise generating functions $\varphi_i(\cdot)$ $(i \in [n_s])$, the algorithm generates and adds corruption to all the sensors

at once. Then the corresponding loss can be computed to update parameters using back-propagation. This algorithm is denoted by TrainASN and tested in experiments to investigate whether the procedure is also able to cover robustness against single source noise.

Minimization of MaxSSN loss

Algorithm 2 TrainSSN

for $i_{\text{iter}} = 1$ to m **do**
  Sample $(y, \{x_i\}_{i=1}^{n_s})$
  if $i_{\text{iter}} \equiv 1 \pmod 2$ **then**
    for $j = 1$ to n_s **do**
      Generate noise $\epsilon_j = \varphi_j(x_j)$
      $\hat{\mathcal{L}}_j^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_j + \epsilon_j, x_{-j}\}))$
    end for
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \max_j \hat{\mathcal{L}}_j^{(i_{\text{iter}})}$
  else
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_i\}_{i=1}^{n_s}))$
  end if
  Update f using $\nabla \mathcal{L}^{(i_{\text{iter}})}$
end for

Minimization of the MaxSSN loss requires n_s (number of input sources) forward-propagations within one iteration. Each propagation needs a different set of corrupted samples generated by adding single source noise to the fixed clean mini-batch of data. There are two possible approaches to compute gradients properly from these multiple passes. First, we can run back-propagation n_s times to save the gradients temporarily without updating any parameters, then the saved gradients with the maximum loss is used for updating parameters. However, this process requires not only n_s forward and backward passes but

also large memory usage proportional to n_s for saving the gradients. Another reasonable approach is to run n_s forward passes to find the maximum loss and compute gradients by going back to the corresponding set of corrupted samples. Algorithm 2 adopts this idea for its efficiency, $n_s + 1$ forward passes and one back-propagation.

Algorithm 3 TrainSSNAlt

for $i_{\text{iter}} = 1$ to m **do**
  Sample $(y, \{x_i\}_{i=1}^{n_s})$
  if $i_{\text{iter}} \equiv 1 \pmod 2$ **then**
    $j \leftarrow (\lfloor i_{\text{iter}}/2 \rfloor \bmod n_s) + 1$
    Generate noise $\epsilon_j = \varphi_j(x_j)$
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_j + \epsilon_j, x_{-j}\}))$
  else
    $\mathcal{L}^{(i_{\text{iter}})} \leftarrow \mathcal{L}(y, f(\{x_i\}_{i=1}^{n_s}))$
  end if
  Update f using $\nabla \mathcal{L}^{(i_{\text{iter}})}$
end for

A faster version of the algorithm, TrainSSNAlt, is also considered since multiple forward passes may take longer as the number of sources increases. This algorithm ignores the maximum loss and alternately augments corrupted data. By a slight abuse of notation, symbols used in our algorithms also represent the iteration steps with the size of mini-batches greater than one. Also, $f(x_1, \cdots, x_{j-1}, x_j + \epsilon_j, x_{j+1}, \cdots, x_{n_s})$ is shortened to $f(\{x_j + \epsilon_j, x_{-j}\})$ in the algorithms.

5.3.2 Feature fusion methods

Fusion of features extracted from multiple input sources can be done in various ways (Chen et al., 2017). One of the popular methods is to fuse via an element-wise mean operation (Ku et al., 2018), but this assumes that each feature must have a same shape, i.e., width, height, and number of channels for a 3D feature. An element-wise mean can be also viewed as averaging channels from different 3D features, and it has an underlying assumption that the channels of each feature should share same information regardless of the input source origin. Therefore, the risk of becoming vulnerable to single source corruption may increase with this simple mean fusion method.

Our fusion method, latent ensemble layer (LEL), is devised for three objectives: (i) maintaining the known advantage—error reduction—of ensemble methods (Tumer and Ghosh, 1996b,a), (ii) admitting source-specific features to survive even after the fusion procedure, and (iii) allowing each source to provide a different number of channels. The proposed layer learns parameters so that channels of the 3D features from the different sources can be selectively mixed. Sparse constraints are introduced to let the training procedure find good subsets of channels to be fused across the n_s feature maps. For example, mixing the i^{th} channel of the convolutional feature from an RGB image with the j^{th} and k^{th} channels of the LIDAR's latent feature is possible in our LEL, whereas in an element-wise mean layer the i^{th} latent channel from RGB is only mixed with the other sources' i^{th} channels. Definition 5.2 explains the details of our LEL, and Figure 5.1 visualizes the overall process.

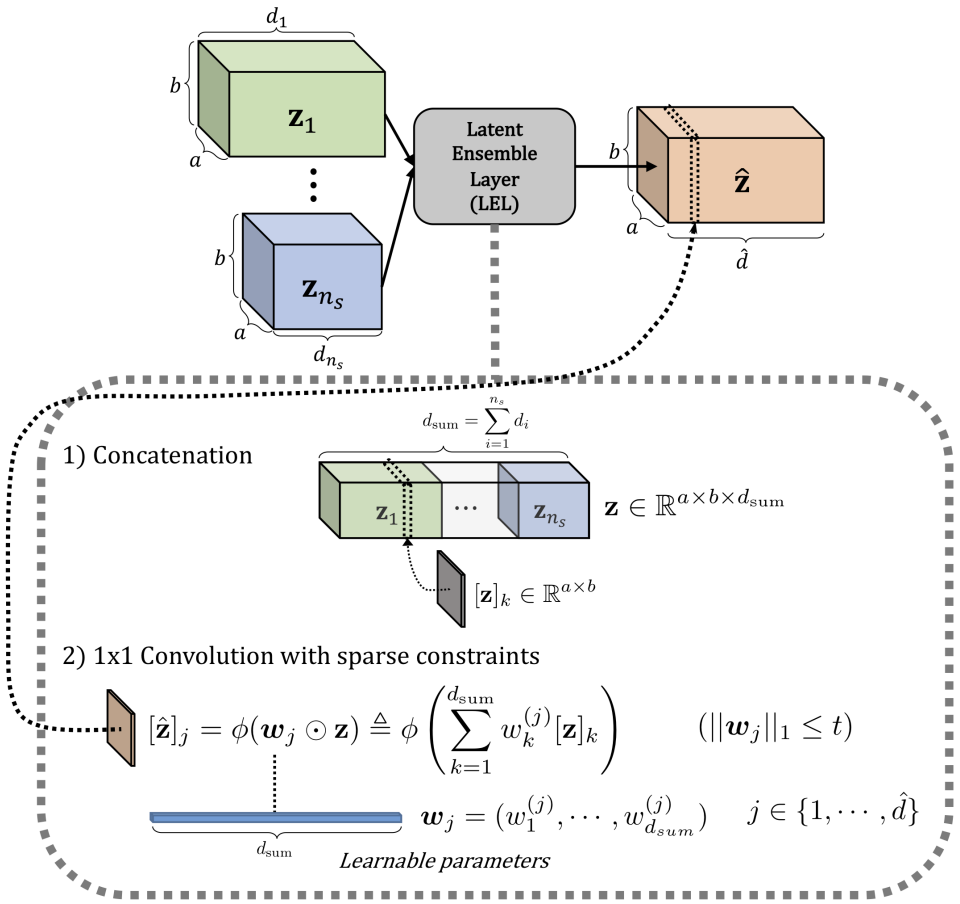


Figure 5.1: Latent ensemble layer (LEL)

Definition 5.2 (Latent ensemble layer)**.** *Suppose we have n_s convolutional features $\mathbf{z}_i \in \mathbb{R}^{a\times b\times d_i}$ from different input sources ($i \in [n_s]$), which can be stacked as $\mathbf{z} = (\mathbf{z}_1, \cdots, \mathbf{z}_m) \in \mathbb{R}^{a\times b\times d_{sum}}$ ($d_{sum} = \sum_{i=1}^{m} d_i$). The k^{th} channel of the stacked feature is denoted by $[\mathbf{z}]_k \in \mathbb{R}^{a\times b}$. Let $\boldsymbol{w}_j = (w_1^{(j)}, \cdots, w_{d_{sum}}^{(j)})$ be a d_{sum}-dimensional weight vector to mix $\mathbf{z}_i$'s in channel-wise fashion. Then LEL outputs $\hat{\mathbf{z}} \in \mathbb{R}^{a\times b\times \hat{d}}$ where each channel is computed as $[\hat{\mathbf{z}}]_j = \phi(\boldsymbol{w}_j \odot \mathbf{z}) \triangleq \phi\left(\sum_{k=1}^{d_{sum}} w_k^{(j)}[\mathbf{z}]_k\right)$, with some activation function ϕ and sparse constraints*

$||\boldsymbol{w}_j||_0 \leq t$ *for all* $j \in \{1, \cdots, \hat{d}\}$.

In practice, this layer can be easily constructed by using 1×1 convolutions with the ReLU activation and ℓ_1 constraints. We also apply an activation function to supplement a semi-adaptive behavior to the fusion procedure. Depth of the output channel is set to $\hat{d} = \max_i\{d_i\}$ and we set the hyper-parameter for ℓ_1 constraint as 0.01 in the experiments.

5.4 Experimental Results

We test our algorithms and the LEL fusion method on 3D and BEV object detection tasks using the car class of the KITTI dataset (Geiger et al., 2012). 3D detection is both an important problem in self-driving cars and one where multiple sensors can contribute fruitfully by providing both complementary and shared information. In contrast, models for 2D object detection heavily rely on RGB data, which typically dominates other modalities. As our experiments include random generation of corruption, each task is evaluated 5 times to compare average scores (reported with 95% confidence intervals), and thus a validation set is used for ease of manipulating data and repetitive evaluation. We follow the split of Ku et al. (2018), 3712 and 3769 frames for training and validation sets, respectively. Results are reported based on three difficulty levels defined by KITTI (easy, medium, hard) and a standard metric Average Precision (AP) is used. A recent open-sourced 3D object detector AVOD (Ku et al., 2018) with a feature pyramid network is selected as a baseline algorithm.

Four different algorithms are compared: AVOD trained on (i) clean data, (ii) data augmented with ASN samples (TrainASN), (iii) SSN augmented data with direct MaxSSN loss minimization (TrainSSN), and (iv) SSN augmented data (TrainSSNAlt). The AVOD architecture is varied to use either element-wise mean fusion layers or our LELs. We follow the original training setups of AVOD, e.g., 120k iterations using an ADAM optimizer with an initial learning rate of 0.0001. Our methods are implemented with TensorFlow on top of the official AVOD code. The computing machine has a Intel Xeon E5-1660v3 CPU with Nvidia Titan X Pascal GPUs. The source code is available at https://github.com/twankim/avod_ssn.

Corruption methods

Our first corruption method, *Gaussian noise* generated i.i.d. with $\mathcal{N}(0, \sigma^2_{\text{Gaussian}})$, is directly added to the pixel value of an image (r, g, b) and the coordinate value of a LIDAR's point (x, y, z). σ_{Gaussian} is set to 0.75τ experimentally with $\tau_{\text{RGB}} = 255$ and $\tau_{\text{LIDAR}} = 0.2$.

The second method *downsampling* selects only 16 out of 64 lasers of LIDAR data. To match this effect, 3 out of 4 horizontal lines of an RGB image are deleted. Effects of corruption on each input source are visualized in Figure 5.2 and 5.3, where the color of a 2D LIDAR image represents a distance from the sensor. Although our analyses in Section 5.2.2 assume the noise variances to be identical, it is nontrivial to set equal noise levels for different modalities in practice, e.g., RGB pixels vs points in a 3D space. Nevertheless, an underlying

(a) Original

(b) Gaussian noise

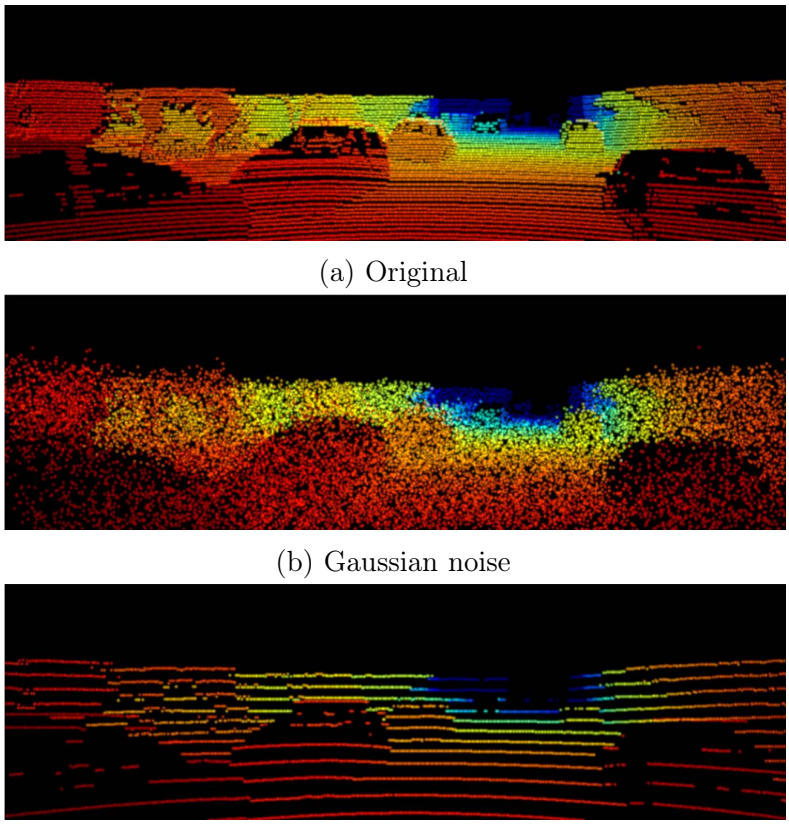
(c) Downsampling

Figure 5.2: Visualization of corrupted RGB image samples

objective of our MaxSSN loss, balancing the degradation rates of different input sources' faults, does not depend on the choice of noise types or levels.

Evaluation metrics for single source robustness

To assess the robustness against single source noise, a new metric minAP is introduced. The AP score is evaluated on the dataset with a single corrupted input source, then after going over all n_s sources, minAP reports the lowest



(a) Original

(b) Gaussian noise

(c) Downsampling

Figure 5.3: Visualization of corrupted LIDAR point cloud samples. The point clouds are projected onto the 2D image plane for easier visual comparison.

score among the n_s AP scores. Our second metric maxDiffAP computes the maximum absolute difference among the scores, which measures the balance of different input sources' single source robustness; low value of maxDiffAP means the well-balanced robustness.

Results

When the fusion model uses the element-wise mean fusion (Table 5.1), TrainSSN algorithm shows the best single source robustness against Gaussian SSN while preserving the original performance on clean data (only small decrease in the moderate BEV detection)[1]. Also a balance of the both input sources' performance is dramatically decreased compared to the models trained without robust learning and a naive TrainASN method.

Encouragingly, AVOD model constructed with our LEL method already achieves relatively high robustness without any robust learning strategies compared to the mean fusion layers. For all the tasks, minAP scores are dramatically increased, e.g., 61.97 vs. 47.41 minAP for the easy 3D detection task, and the maxDiffAP scores are decreased (maxDiffAP scores for AVOD with LEL are reported in Appendix A.2.). Then the robustness is further improved by minimizing our MaxSSN loss. As our LEL's structure inherently handles corruption on a single source well, even the TrainASN algorithm can successfully guide the model to be equipped with the desired robustness.

A corruption method with a different style, downsampling, is also tested with our LEL fusion method. Table 5.2 shows that the model trained with our TrainSSN algorithm achieves the best robustness among the four algorithms for this complex and realistic perturbation.

[1] In practice, it is difficult to identify flexible parameters related to shared information in advance, and also the design goal becomes a soft rather than a hard constraint. Therefore there is minor degradation in performance, to pay for the added robustness.

Table 5.1: Car detection (3D/BEV) perrkformance of AVOD with *element-wise mean* fusion layers and *latent ensemble layers (LEL)* against *Gaussian* SSN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
			***Fusion method**: Mean*			
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	**76.41**	**72.74**	**66.86**	89.33	86.49	**79.44**
+TrainASN	75.96	66.68	65.97	88.63	79.45	78.79
+TrainSSN	76.28	67.10	66.51	88.86	79.60	79.11
+TrainSSNAlt	**77.46**	67.61	66.06	**89.68**	**86.71**	79.41
(Gaussian SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	47.41±0.28	41.84±0.17	36.47±0.16	65.63±0.28	58.02±0.23	50.43±0.14
+TrainASN	61.53±0.57	52.72±0.08	47.25±0.13	87.71±0.14	78.37±0.06	77.85±0.08
+TrainSSN	71.65±0.31	**62.14±0.08**	**56.78±0.12**	88.21±0.08	78.90±0.09	**77.92±0.11**
+TrainSSNAlt	**71.66±0.48**	57.61±0.12	55.90±0.11	**89.42±0.04**	**79.56±0.06**	**77.92±0.05**
(Gaussian SSN)	$\max \text{Diff} AP_{3D}(\%)$			$\max \text{Diff} AP_{BEV}(\%)$		
AVOD	26.70±0.52	22.42±0.29	20.92±0.25	22.27±0.41	20.76±0.33	20.09±0.20
+TrainASN	14.48±0.82	12.72±0.33	11.18±0.27	0.88±0.22	0.48±0.13	0.28±0.12
+TrainSSN	**3.71±0.46**	**3.42±0.25**	7.50±0.25	0.36±0.17	**0.04±0.15**	0.71±0.17
+TrainSSNAlt	5.55±0.81	8.73±0.32	**2.91±0.22**	**0.09±0.14**	0.13±0.11	**0.18±0.11**
			***Fusion method**: Latent Ensemble Layer*			
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	**77.79**	**67.69**	**66.31**	**88.90**	**85.64**	**78.86**
+TrainASN	75.00	64.75	58.28	88.30	78.60	77.23
+TrainSSN	74.25	65.00	63.83	87.88	78.84	77.66
+TrainSSNAlt	76.04	66.42	64.41	88.80	79.53	78.53
(Gaussian SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	61.97±0.55	53.95±0.42	47.24±0.27	79.44±0.09	72.46±3.14	68.25±0.06
+TrainASN	**74.24±0.38**	58.25±0.16	**56.13±0.10**	88.10±0.26	**78.19±0.13**	70.42±0.07
+TrainSSN	68.16±0.88	**60.39±0.38**	56.04±0.28	**88.12±0.16**	78.17±0.06	70.21±0.05
+TrainSSNAlt	68.63±0.40	55.48±0.16	54.42±0.17	86.51±0.46	76.85±0.11	**71.95±2.72**

Remark 5.3. *A simple* TrainSSNAlt *achieves fairly robust models in both fusion methods against Gaussian noise, and two reasons may explain this phenomenon. First, all parameters are updated instead of fine-tuning only fusion related parts. Therefore, unlike our analyses on the linear model, the latent representation can be transformed to meet the objective function. In fact,* TrainSSNAlt *performs poorly when we fine-tune the model with concatenation fusion layers as shown in the supplement. Secondly, the loss function $\mathcal{L}$ inside our $\mathcal{L}_{\text{MaxSSN}}$ is usually non-convex so that it may be enough to use an indirect*

Table 5.2: Car detection (3D/BEV) performance of AVOD with *latent ensemble layers (LEL)* against *downsampling* SSN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	**77.79**	**67.69**	**66.31**	88.90	**85.64**	**78.86**
+TrainASN	71.74	61.78	60.26	87.29	77.08	75.89
+TrainSSN	75.54	66.26	63.72	88.07	79.18	78.03
+TrainSSNAlt	76.22	66.05	63.87	**89.00**	79.65	78.03
(Downsample SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	61.70	51.66	46.17	86.08	69.99	61.55
+TrainASN	65.74	53.49	51.35	82.27	67.88	65.79
+TrainSSN	**73.33**	**57.85**	**54.91**	**86.61**	**76.07**	**68.59**
+TrainSSNAlt	64.77	53.34	48.29	85.27	69.87	67.77

approach for small number of sources, $n_s = 2$.

5.5 Conclusion

We study two strategies to improve robustness of fusion models against single source corruption. Motivated by analyses on linear fusion models, a loss function is introduced to balance performance degradation of deep fusion models caused by corruption in different sources. We also demonstrate the importance of a fusion method's structure by proposing a simple ensemble layer achieving such robustness inherently. Our experimental results show that deep fusion models can effectively use complementary and shared information of different input sources by training with our loss and fusion layer to obtain both robustness and high accuracy. We hope our results motivate further work to improve the single source robustness of more complex fusion models with either large number of input sources or adaptive networks. Another interesting

direction is to investigate the single source robustness against adversarial attacks in deep fusion models, which can be compared with our analyses in the supplementary material.

Chapter 6

Conclusion and Future Work

6.1 Summary of the Dissertation

Deep learning algorithms for visual data and fusion methods for multiple input sources are known to be successful strategies to achieve superior performance in machine learning. CNN-based object detection models and various sensors like cameras, radar, and LIDAR are being actively investigated for robust perception systems. This dissertation addresses the problem of combining two effective methods to develop robust deep fusion models for practical applications in autonomous cars. In particular, we focused on two objectives in learning deep fusion based object detection models: (1) overcoming inherent limitations of a single sensor framework, and (2) equipping robustness against corruption in a single input source.

CLF2 R-CNN, covered in Chapter 3, is one of the earliest deep fusion based road user detection models relying on both RGB and LIDAR data. To effectively utilize LIDAR's robustness against bad illumination conditions and region-based CNN models' superb detection performance on visual data, our model transforms LIDAR's 3D point clouds to 2D depth images and extracts additional region proposals. The results on KITTI object detection benchmark

illustrate the benefit of having complementary object proposals extracted from the different types of sensors; objects missed by an RGB-only framework are successfully recovered by our fusion approach.

We next propose a deep fusion model **SSDFusion** in Chapter 4 to overcome a single sensor's functional limit by learning a joint representation from RGB and LIDAR sensors for multiple learning tasks. Our single stage-based fusion model detects pedestrians and simultaneously estimates the distance values in real-time due to our simplified depth prediction task. To train the model for our specific tasks, we collected a new pedestrian detection dataset **SvDPed** with single valued distance labels using practical sensors including a low-resolution LIDAR with 16 lasers. Our collection method obtains ground truth depth labels from GNSS and camera poses rather than other on-vehicle range sensors to minimize an overlapping usage of input sources and to gather accurate data even for long ranges.

Finally, Chapter 5 discusses robustness of deep fusion models against single source faults considering shared information in multiple input sources. Studies on our simple linear data fusion model reveals the necessity of supplementary strategy for the desired robustness. We first provide effective training algorithms for our novel **MaxSSN** loss which is designed to balance vulnerabilities of different sensors. Then we introduce a convolutional fusion layer, **LEL**, to deal with single source noise using structural benefits of ensemble methods in a latent space. Both approaches are applied to a deep fusion based 3D object detector and extensively evaluated on KITTI dataset using an RGB

image and LIDAR point clouds.

6.2 Future Work

6.2.1 Deep Fusion Models and Adversarial Attacks

A recently found important factor for trustworthy models is robustness against *adversarial attacks* (Biggio et al., 2013; Szegedy et al., 2014; Goodfellow et al., 2015). Interestingly, a small perturbation added to the input image of a trained model can cause a wrong prediction with high confidence, even if the difference is imperceptible to a human. This has become a critical issue, because even high performance models are shown to be vulnerable to adversaries, which degrades the reliability. As technologies are moving forward to have connectivity with high speed and high data rate communication systems, e.g. 5G cellular networks, adversarial attacks can be applied remotely to cause a serious malfunctioning of autonomous vehicles. Effective usage of multiple sensors is expected to support the model's defense capacity against such attacks. For instance, if one sensor is perturbed by an adversary but other sensors with correlated information are not attacked, the total level of confusion can be decreased to have better model resilience against such perturbations. Both analytical and experimental studies on our MaxSSN loss and the LEL layer highlight the promise of our approaches, and extending the studies to complex adversarial perturbation would be a compelling direction for future research.

6.2.2 Explainable Deep Fusion Models

Explaining the inherent decision making process of a complex machine learning model is another principal research topic for building a trustworthy system. Deep learning directly extracts rich features from raw data like images. Although these features lead to a high predictive performance, it is difficult to interpret the representations or the logic of the system in a human-friendly way. Sensor-fusion models are expected to achieve higher performance and robustness by effectively incorporating each sensor's information. However, there is no guarantee that the trained fusion model is making decisions with proper reasons. For example, the contribution of an RGB camera to object detection in a dark or foggy day should be lower than a sunny day, when the model should rely more on other complementary sensors. Although the model outputs a probability score for each class per each object, this cannot explain the reasoning like the aforementioned importance per sensor. Applying existing interpretation methods for deep learning (Zhou et al., 2016; Selvaraju et al., 2017; Bach et al., 2015), to deep fusion models can be a simple and meaningful starting point. Further, developing deep fusion models that can directly output the contribution of each sensor to the model's prediction without going through any additional backward propagation steps will make the real-time examination of deployed models feasible.

Appendix

Appendix

Additional Results for Chapter 5

A.1 Single Source Adversarial Attacks

Another important type of perturbation is an adversarial attack. Different from the previously studied random noise, perturbation to the input sources is also optimized to maximize the loss to consider the worst case. Adversarial version of the MaxSSN loss is defined as follows:

Definition A.1. *For multiple sources $x_1, \cdots, x_{n_s}$ and a target variable y, denote a predefined loss function by $\mathcal{L}$. If each input source x_i is maximally perturbed with some additive noise $\eta_i \in \mathcal{S}_i$ for $i \in [n_s]$,* AdvMaxSSN *loss for a model f is defined as follows:*

$$\mathcal{L}_{\text{AdvMaxSSN}}(f, \eta) \triangleq \max_i \left\{ \max_{\eta_i \in \mathcal{S}_i} \mathcal{L}\left(y, f(x_i + \eta_i, x_{-i})\right) \right\}_{i=1}^{n_s}$$

As a simple model analysis, let's consider a binary classification problem using the logistic regression. Again, two input sources $x_1 = [z_1; z_3]$ and $x_2 = [z_2; z_3]$ have a common feature vector z_3 as in the linear fusion data model. A binary classifier $\text{sgn}(f(x_1, x_2))$ is trained to predict label $y \in \{-1, 1\}$, where $f(x_1, x_2) = (w_1^T z_1 + g_1^T z_3) + (w_2^T z_2 + g_2^T z_3)$ and the training loss is $\mathbb{E}_{x,y}\left[\ell\left(y \cdot f(x_1, x_2)\right)\right]$ with the logistic function $\ell(x) = \log(1 + \exp(-x))$. Here,

we apply one of the most popular attacks, fast gradient sign (FGS) method, which was also motivated by linear models without a fusion framework (Goodfellow et al., 2015). The adversarial attack η_i per each source x_i under ℓ_∞ norm constraint $||\eta_i||_\infty \leq \varepsilon$ can be similarly derived as follows:

$$\begin{aligned} \eta_1 &= [-\varepsilon y \cdot \text{sgn}(w_1); -\varepsilon y \cdot \text{sgn}(g_1)], \\ \eta_2 &= [-\varepsilon y \cdot \text{sgn}(w_2); -\varepsilon y \cdot \text{sgn}(g_2)] \end{aligned} \tag{A.1}$$

As a substitute for the linear fusion data model, let's assume the true classes are generated by the hidden relationship $y = \text{sgn}\left(\sum_{i=1}^{3} \beta_i^T z_i\right)$. Then the optimal fusion binary classifier becomes $\text{sgn}(f_{\text{direct}}(x_1, x_2))$. Similar to the previous section, suppose an objective is to find a model with robustness against single source adversarial attacks, while preserving the performance on clean data. Then the overall optimization problem can be reduced to the following one:

$$\begin{aligned} &\min_{g_1, g_2} \max \left\{ \mathcal{L}\left(y, f_{\text{direct}}(x_1 + \eta_1, x_2)\right), \mathcal{L}\left(y, f_{\text{direct}}(x_1, x_2 + \eta_2)\right) \right\} \\ &\qquad s.t. \;\; g_1 + g_2 = \beta_3 \end{aligned} \tag{A.2}$$

As ℓ is a decreasing function, optimal g_1 and g_2 of the original problem are equivalent to the minimizer of the following one:

$$\begin{aligned} &\varepsilon \min_{g_1, g_2} \max \left\{ ||w_1||_1 + ||g_1||_1, ||w_2||_1 + ||g_2||_1 \right\} \\ &\qquad s.t. \;\; g_1 + g_2 = \beta_3 \end{aligned} \tag{A.3}$$

By solving this convex optimization problem, we can achieve $\mathcal{L}^*_{\text{AdvMaxSSN}}$ and optimizers g_1^*, g_2^*. Also, we can find $\mathcal{L}'_{\text{AdvMaxSSN}}$, a $\mathcal{L}_{\text{AdvMaxSSN}}$ value evaluated

using the optimal model for minimizing the adversarial attacks added to all the sources at once.

Interestingly, we can show that:

$$\mathcal{L}'_{\text{AdvMaxSSN}} \geq \mathcal{L}^*_{\text{AdvMaxSSN}} \qquad \text{if } \frac{||\beta_2||_1 - ||\beta_1||_1}{||\beta_3||_1} > 1$$
$$\mathcal{L}'_{\text{AdvMaxSSN}} = \mathcal{L}^*_{\text{AdvMaxSSN}} \qquad \text{Otherwise}$$

In other words, if inherent influence of z_1 and z_2 are well balanced compared to the common feature z_3 in the sense of ℓ_1 norm, adversarial attacks only applied to a single source can be equivalently defended by just using a traditional adversarial training strategy to learn a model robust against attacks added to all the sources at once.

Proof. The original minimizing $\mathcal{L}_{\text{AdvMaxSSN}}$ loss minimization problem with an additional constraint of preserving loss under clean data can be transformed to the problem stated in (A.1) due to the flexibility of g_1 and g_2:

$$\min_{g_1,g_2} \max \left\{ \mathcal{L}\left(y, f_{\text{direct}}(x_1 + \eta_1, x_2)\right), \mathcal{L}\left(y, f_{\text{direct}}(x_1, x_2 + \eta_2)\right) \right\}$$
$$s.t. \;\; g_1 + g_2 = \beta_3$$

As η_i's are assumed to be made with FGS method, adversarial attacks under ℓ_∞ norm constraints are as follows:

$$\eta_1 = [-\varepsilon y \cdot \text{sgn}(w_1); -\varepsilon y \cdot \text{sgn}(g_1)], \;\; \eta_2 = [-\varepsilon y \cdot \text{sgn}(w_2); -\varepsilon y \cdot \text{sgn}(g_2)]$$

Therefore, minimizing $\mathcal{L}_{\text{AdvMaxSSN}}(f_{\text{direct}}, \eta)$ over g_1, g_2 becomes:

$$\min_{g_1,g_2} \max\{\mathbb{E}\left[\ell\left(y \cdot f_{\text{direct}}(x_1, x_2) - \varepsilon(||w_1||_1 + ||g_1||_1)\right)\right], \\ \mathbb{E}\left[\ell\left(y \cdot f_{\text{direct}}(x_1, x_2) - \varepsilon(||w_2||_1 + ||g_2||_1)\right)\right]\} \quad s.t. \;\; g_1 + g_2 = \beta_3$$

We can solve the following problem to find minimizers g_1^* and g_2^*.

$$\min_{g_1,g_2} \max\left\{||w_1||_1 + ||g_1||_1, ||w_2||_1 + ||g_2||_1\right\} \quad s.t. \;\; g_1 + g_2 = \beta_3$$

Similar to the random noise case, substitute variables as $g = g_1, v = \beta_3, c_1 = ||\beta_1||_1, c_2 = ||\beta_2||_2$, and solve the following convex optimization problem:

$$\min_{g} \max\{||g||_1 + c_1, ||g - v||_1 + c_2\}$$

which can be solved by introducing γ,

$$\min_{g,\gamma} \gamma \quad s.t. \;\; c_1 + ||g||_1 - \gamma \leq 0, \;\; c_2 + ||g - v||_1 - \gamma \leq 0$$

KKT condition gives:

$$\begin{aligned}
&\text{(Primal feasibility)} && c_1 + ||g||_1 - \gamma \leq 0, \;\; c_2 + ||g - v||_1 - \gamma \leq 0 \\
&\text{(Dual feasibility)} && \lambda_1 \geq 0, \;\; \lambda_2 \geq 0 \\
&\text{(Complementary slackness)} && \lambda_1(c_1 + ||g||_1 - \gamma) = 0, \\
& && \lambda_2(c_2 + ||g - v||_1 - \gamma) = 0 \\
&\text{(Stationary)} && \lambda_1 + \lambda_2 = 1, \;\; 0 \in \lambda_1 \partial||g||_1 + \lambda_2 \partial||g - v||_1
\end{aligned}$$

If $\lambda_1 = 0$ or $\lambda_2 = 0$, these cases handle when the inherent imbalance of three components z_1, z_2 and z_3. Consider $\lambda_2 = 0$, which gives $||g||_1 + c_1 - \gamma = 0$

from the complementary slackness condition. And the stationary condition becomes $0 \in \partial||g||_1$. As a subgradient of $||g||_1$ can be zero if and only if $g(i) = 0$ for any i^{th} component, the solution is $g = 0$ with $\gamma = c_1$ and the necessary condition is $||v||_1 + c_2 \leq c_1$. Similar solution can be found for $\lambda_1 = 0$ case as $g = v, \gamma = c_2$ if $||v||_1 + c_1 \leq c_2$. Therefore, we can have $\gamma^* = \min\max\{||w_1||_1 + ||g_1||_1, ||w_2||_1 + ||\beta_3 - g_1||_1\}$ and corresponding parameters as:

$$(\gamma^*, g_1^*, g_2^*) = \begin{cases} (||\beta_2||_1, \beta_3, 0) & \text{if } ||\beta_1||_1 + ||\beta_3||_1 \leq ||\beta_2||_1 \\ (||\beta_1||_1, 0, \beta_3) & \text{if } ||\beta_2||_1 + ||\beta_3||_1 \leq ||\beta_1||_1 \end{cases}$$

Now let's consider $\lambda_1 \neq 0, \lambda_2 \neq 0$. Denote $q \in \lambda_1 \partial||g||_1 + \lambda_2 \partial||g - v||_1$ as the element of subdifferential of the Lagrangian. We need to find cases for $q(i) = 0$ to hold.

(i) If $v(i) = 0$, then $\text{sgn}(g(i)) = \text{sgn}(g(i) - v(i))$ holds. Therefore, if $g(i) \neq= 0$, a subgradient becomes $q(i) = \lambda_1 \text{sgn}(g(i)) + \lambda_2 \text{sgn}(g(i)) = \text{sgn}(g(i))$ which cannot be zero. $\Rightarrow g(i) = v(i) = 0$.

(ii) If $v(i) \neq 0$, we need to consider three different sub cases.

First, if $g(i) \neq 0$ and $g(i) \neq v(i)$, then $q(i) = \lambda_1(\text{sgn}(g(i)) - \text{sgn}(g(i) - v(i))) + \text{sgn}(g(i) - v(i))$. For $q(i) = 0$ to hold, $\text{sgn}(g(i)) = -\text{sgn}(g(i) - v(i))$ must be true with $\lambda_1 = \frac{1}{2}$. This gives a solution $g(i) = \alpha_i v(i)$ with $\forall \alpha_i \in (0, 1)$.

Secondly, if $g(i) = 0$ but $g(i) \neq v(i)$, then the subgradient is $q(i) = \lambda_1 \alpha_i + (1 - \lambda_1)\text{sgn}(-v(i))$ for any $\alpha_i \in [-1, 1]$. Therefore, if $\alpha_i = \frac{1-\lambda_1}{\lambda_1}\text{sgn}(v(i))$

with some $\lambda_1 \in [\frac{1}{2}, 1)$, the stationary condition holds.

Finally, if $g(i) \neq 0$ and $g(i) = v(i)$, then $q(i) = \lambda_1 \text{sgn}(g(i)) + (1 - \lambda_1)\alpha_i$ for any $\alpha_i \in [-1, 1]$. Therefore, if $\alpha_i = \frac{\lambda_1}{1-\lambda_1}\text{sgn}(v(i))$ with $\lambda_1 \in (0, \frac{1}{2}]$, $q(i) = 0$ holds for the stationary condition.

All the above cases in (i) and (ii) can be restated as a combined solution $g(i) = \alpha_i v(i)$, $\forall \alpha_i \in [0, 1]$. It is easy to show that $|g(i)| + |g(i) - v(i)| = |v(i)|$ holds for any i. Also, $\lambda_1 \neq 0, \lambda_2 \neq 0$ with the complementary slackness condition gives a new constraint $\gamma = ||g||_1 + c_1 = ||g - v||_1 + c_2$. Hence, we can calculate γ by averaging the two equivalent values:

$$\gamma = \frac{1}{2}(c_1 + c_2 + ||g||_1 + ||g - v||_1) = \frac{1}{2}(c_1 + c_2 + ||v||_1)$$

Therefore,

$$(\gamma^*, g_1^*, g_2^*) = \left(\frac{1}{2}(||\beta_1||_1 + ||\beta_2||_1 + ||\beta_3||_1), \alpha \odot \beta_3, \beta_3 - \alpha \odot \beta_3\right),$$

where $\odot$ is an element-wise product and each element of α can have any value in $[0, 1]$, i.e. $\alpha(i) \in [0, 1]$.

Now, let's consider a model robust against adversarial attacks added to both sources x_1 and x_2 at the same time. This becomes a problem of minimizing $||\beta_1||_1 + ||\beta_2||_1 + ||g_1||_1 + ||\beta_3 - g_1||_1$. And the optimal solution can be achieved by $(g_1', g_2') = (\alpha \odot \beta_3, \beta_3 - \alpha \odot \beta_3)$ for any alpha satisfying $\alpha(i) \in [0, 1]$. Therefore, we can conclude that our $\mathcal{L}_{\text{AdvMaxSSN}}$ loss is necessary to give a binary classifier more robust against single source adversarial attacks,

i.e. $\mathcal{L}^*_{\text{AdvMaxSSN}} \leq \mathcal{L}'_{\text{AdvMaxSSN}}$, if $\frac{||\beta_2||_1 - ||\beta_1||_1}{||\beta_3||_1} > 1$ holds. Surprisingly, if $\frac{||\beta_2||_1 - ||\beta_1||_1}{||\beta_3||_1} \leq 1$ holds to have balanced influence from inherent components from the different source of inputs, $\mathcal{L}^*_{\text{AdvMaxSSN}} = \mathcal{L}'_{\text{AdvMaxSSN}}$. In other words, if different input sources contributes to the target variable with certain balance, a traditional way of generating adversarial samples by considering all the sources at once can train a model robust against single source attacks as well. □

A.2 Additional Experimental Results

Evaluation on ASN data

Although our main focus is corruption on a single source, it is possible for a model to encounter a case where all the sources are corrupted. If the level of corruption is severe, then extracting any meaningful information from the input sources is impossible, e.g. occlusion on every sensors. However, we hope our model to be robust against reasonably corrupted input sources even if our training objective leans toward the single source robustness. Therefore, we also report the model's performance against data corrupted with ASN. In most cases, the AVOD learned with TrainASN method achieves the best robustness against ASN, which is designed to do so. However, a model using element-wise mean fusion layers trained with TrainASN shows lower robustness scores compared to the SSN oriented approaches. We believe that this phenomenon is caused by corrupted feature extraction combined with the structural limitation of the mean fusion layer.

Fine-tuning

We also consider another algorithmic framework using *fine-tuing*. The algorithm starts with a normal training on clean data for m_{clean} iterations, which may include some general data augmentation methods like random cropping, and flipping. Then m_{tune} steps of fine-tuning is run to update only a subset of the model's parameters, $\boldsymbol{\theta}_{\text{fusion}} \subset f$, so that any essential parts for extracting features from normal data are not affected. Convolutional layers extracting features from different sources before the fusion stages are fixed, and other layers for fusing the features and making predictions are updated in the fine-tuning stage. The experimental results using this method are provided in Table A.2 and A.4 for the Gaussian noise case. Overall performance of the fusion model trained from the scratch is better than using fine-tuning. This shows the importance of feature extraction parts in deep learning models.

Concatenation

Our analyses in Section 5.2 assume to use a linear fusion model with a simple concatenation strategy. Therefore, we first train the AVOD model with concatenation fusion layers on clean data and fine-tune with different training strategies. Interestingly, a simple data augmentation strategy TrainSSNAlt does not work well in this case, and TrainASN algorithm learns the best robust model. Unlike our simple linear model deep learning jointly learns both feature representation and weights for the fusion layers. Also, concatenated convolutional features have large number of channels which are mixed without

Table A.1: Car detection (3D/BEV) performance of AVOD with *element-wise mean* fusion layers against *Gaussian* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	76.41	**72.74**	**66.86**	89.33	86.49	**79.44**
+TrainASN	75.96	66.68	65.97	88.63	79.45	78.79
+TrainSSN	76.28	67.10	66.51	88.86	79.60	79.11
+TrainSSNAlt	**77.46**	67.61	66.06	**89.68**	**86.71**	79.41
(Gaussian ASN)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	28.08±0.91	26.35±2.18	21.81±0.63	42.01±0.23	33.68±0.17	33.60±0.13
+TrainASN	61.26±0.45	47.71±0.24	45.60±0.19	87.40±0.07	72.07±2.89	70.13±0.05
+TrainSSN	69.33±0.43	55.41±0.21	**52.90±2.12**	**88.39±0.13**	**78.37±0.10**	**70.75±0.05**
+TrainSSNAlt	**71.63±0.04**	**56.24±0.16**	49.14±0.10	87.95±0.08	77.88±0.17	69.96±0.08
(Gaussian SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	47.41±0.28	41.84±0.17	36.47±0.16	65.63±0.28	58.02±0.23	50.43±0.14
+TrainASN	61.53±0.57	52.72±0.08	47.25±0.13	87.71±0.14	78.37±0.06	77.85±0.08
+TrainSSN	71.65±0.31	**62.14±0.08**	**56.78±0.12**	88.21±0.08	78.90±0.09	**77.92±0.11**
+TrainSSNAlt	**71.66±0.48**	57.61±0.12	55.90±0.11	**89.42±0.04**	**79.56±0.06**	**77.92±0.05**
(Gaussian SSN)	$\max \text{Diff} AP_{3D}(\%)$			$\max \text{Diff} AP_{BEV}(\%)$		
AVOD	26.70±0.52	22.42±0.29	20.92±0.25	22.27±0.41	20.76±0.33	20.09±0.20
+TrainASN	14.48±0.82	12.72±0.33	11.18±0.27	0.88±0.22	0.48±0.13	0.28±0.12
+TrainSSN	**3.71±0.46**	**3.42±0.25**	7.50±0.25	0.36±0.17	**0.04±0.15**	0.71±0.17
+TrainSSNAlt	5.55±0.81	8.73±0.32	**2.91±0.22**	**0.09±0.14**	0.13±0.11	**0.18±0.11**

sparse constraints. Therefore, this may lead to a model with too complex joint feature representation which needs stronger guideline in optimization steps.

Results on downsampling corruption

Downsampling the LIDAR sensor is important as it is not clear whether a model trained with a high-resolution sensor will still work with a low-resolution one. In fact, reducing the number of lasers of a LIDAR is directly related to its price, which an important practical issue in deploying an actual autonomous vehicle. As the rotating LIDAR sensor used in the KITTI dataset outputs point clouds with a horizontal structure, an RGB image's horizontal lines are also set to black to match the information loss ratio 1/4. Table A.6 fully reports

Table A.2: Car detection (3D/BEV) performance of AVOD with *element-wise mean* fusion layers (trained with fine-tuning) against *Gaussian* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	**76.41**	**72.74**	**66.86**	**89.33**	**86.49**	**79.44**
+TrainASN	62.55	55.81	55.34	79.08	69.90	69.83
+TrainSSN	73.50	65.66	64.74	88.27	85.65	78.98
+TrainSSNAlt	75.76	71.99	66.31	88.76	85.73	79.14
(Gaussian ASN)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	28.08±0.91	26.35±2.18	21.81±0.63	42.01±0.23	33.68±0.17	33.60±0.13
+TrainASN	**68.58±1.93**	**54.76±0.30**	**48.00±0.29**	**83.15±3.01**	**76.10±0.069**	**68.49±0.08**
+TrainSSN	60.73±0.32	45.52±0.19	44.42±0.11	78.24±0.10	68.41±0.10	60.45±0.07
+TrainSSNAlt	53.25±0.27	44.96±0.14	38.64±0.10	68.69±0.18	59.41±0.14	51.37±0.07
(Gaussian SSN)		$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$	
AVOD	47.41±0.28	41.84±0.17	36.47±0.16	65.63±0.28	58.02±0.23	50.43±0.14
+TrainASN	52.72±0.34	45.66±0.24	39.29±0.22	69.33±0.21	60.19±0.15	59.66±0.15
+TrainSSN	62.46±0.48	53.85±0.22	47.62±0.14	77.77±0.16	68.71±0.09	67.89±0.09
+TrainSSNAlt	**70.09±0.46**	**56.20±0.21**	**54.46±0.13**	**84.46±2.66**	**76.32±0.06**	**68.74±0.08**

the performance of AVOD using our LEL when downsampling is considered as a corruption method.

Table A.3: Car detection (3D/BEV) performance of AVOD with *latent ensemble layers (LEL)* against *Gaussian* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	**77.79**	**67.69**	**66.31**	**88.90**	**85.64**	**78.86**
+TrainASN	75.00	64.75	58.28	88.30	78.60	77.23
+TrainSSN	74.25	65.00	63.83	87.88	78.84	77.66
+TrainSSNAlt	76.04	66.42	64.41	88.80	79.53	78.53
(Gaussian ASN)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	46.79±0.37	41.46±0.27	36.31±0.20	77.40±0.34	67.46±0.11	59.53±0.11
+TrainASN	**74.24±0.29**	**63.47±0.18**	**57.25±0.19**	87.72±0.12	**77.89±0.09**	**70.36±0.05**
+TrainSSN	67.69±0.28	55.74±0.30	53.16±0.32	**87.73±0.16**	77.80±0.15	70.00±0.10
+TrainSSNAlt	63.72±0.40	53.15±0.29	48.17±0.22	85.36±0.08	75.60±0.08	69.17±0.03
(Gaussian SSN)		min $AP_{3D}(\%)$			min $AP_{BEV}(\%)$	
AVOD	61.97±0.55	53.95±0.42	47.24±0.27	79.44±0.09	72.46±3.14	68.25±0.06
+TrainASN	**74.24±0.38**	58.25±0.16	**56.13±0.10**	88.10±0.26	**78.19±0.13**	70.42±0.07
+TrainSSN	68.16±0.88	**60.39±0.38**	56.04±0.28	**88.12±0.16**	78.17±0.06	70.21±0.05
+TrainSSNAlt	68.63±0.40	55.48±0.16	54.42±0.17	86.51±0.46	76.85±0.11	**71.95±2.72**
(Gaussian SSN)		max Diff$AP_{3D}(\%)$			max Diff$AP_{BEV}(\%)$	
AVOD	3.75±2.05	0.98±0.55	5.95±0.40	7.28±0.37	4.46±3.25	**1.25±0.13**
+TrainASN	**1.54±0.40**	**0.85±0.24**	**0.83±0.25**	0.92±0.17	1.09±0.14	7.44±0.08
+TrainSSN	4.61±1.16	2.51±0.50	0.74±0.46	**0.16±0.32**	**0.72±0.14**	7.10±0.14
+TrainSSNAlt	4.65±1.04	7.88±0.46	2.90±0.45	1.12±0.71	1.83±0.17	3.42±2.84

Table A.4: Car detection (3D/BEV) performance of AVOD with *latent ensemble layers (LEL)* (trained with fine-tuning) against *Gaussian* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	**77.79**	**67.69**	**66.31**	**88.90**	**85.64**	78.86
+TrainASN	74.65	65.40	63.40	88.18	79.21	78.42
+TrainSSN	76.95	67.22	65.66	88.77	79.74	**78.96**
+TrainSSNAlt	76.81	67.46	66.12	88.47	79.62	78.86
(Gaussian ASN)		$AP_{3D}(\%)$			$AP_{BEV}(\%)$	
AVOD	46.79±0.37	41.46±0.27	36.31±0.20	77.40±0.34	67.46±0.11	59.53±0.11
+TrainASN	**63.73±0.24**	**53.16±0.16**	**47.79±0.17**	**80.18±0.07**	**76.26±0.03**	**69.12±0.04**
+TrainSSN	60.80±0.48	47.73±0.13	45.67±0.15	79.82±0.22	69.66±0.10	68.38±0.10
+TrainSSNAlt	52.25±1.47	43.77±0.62	37.91±0.48	77.51±0.12	67.32±0.09	59.65±0.10
(Gaussian SSN)		min $AP_{3D}(\%)$			min $AP_{BEV}(\%)$	
AVOD	61.97±0.55	53.95±0.42	47.24±0.27	79.44±0.09	72.46±3.14	68.25±0.06
+TrainASN	**68.08±0.44**	**57.28±0.18**	**55.27±0.20**	86.45±0.08	77.19±0.08	69.57±0.08
+TrainSSN	67.98±1.31	55.61±0.23	53.76±0.20	**86.87±0.12**	**77.56±0.05**	**69.81±0.08**
+TrainSSNAlt	62.76±0.41	52.14±0.26	46.55±0.13	85.34±2.36	75.72±0.04	68.60±0.02

Table A.5: Car detection (3D/BEV) performance of AVOD with *concatenation* fusion layers (trained with fine-tuning) against *Gaussian* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	**78.40**	**74.88**	**67.78**	**89.74**	**87.76**	**79.83**
+TrainASN	72.89	63.47	62.22	88.44	84.97	78.88
+TrainSSN	76.15	66.79	65.78	89.02	86.06	79.29
+TrainSSNAlt	76.46	72.98	66.94	89.07	86.39	79.34
(Gaussian ASN)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	16.50±2.27	15.12±0.06	15.06±0.08	25.81±0.23	25.38±0.18	17.45±0.08
+TrainASN	**69.21±0.24**	**54.85±0.08**	**53.30±0.08**	**86.07±0.11**	**76.42±0.04**	**69.54±0.02**
+TrainSSN	62.05±0.36	50.35±2.58	46.04±0.25	79.21±0.08	69.31±0.10	61.21±0.06
+TrainSSNAlt	33.86±2.85	27.99±0.64	22.59±0.60	42.65±0.18	41.77±0.18	34.13±0.12
(Gaussian SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	31.23±0.31	30.27±0.13	30.49±0.18	43.04±0.16	42.81±0.10	42.96±0.08
+TrainASN	**68.21±0.37**	54.50±0.26	47.91±0.21	**86.66±0.11**	**76.95±0.11**	**69.70±0.08**
+TrainSSN	64.39±0.23	**55.12±0.21**	**48.38±0.14**	79.71±0.07	70.05±0.07	69.32±0.10
+TrainSSNAlt	44.25±0.49	37.23±0.44	37.58±0.34	59.06±0.12	51.19±0.08	51.28±0.06

Table A.6: Car detection (3D/BEV) performance of AVOD with *latent ensemble layers (LEL)* against *downsampling* SSN and ASN on the KITTI validation set.

(Data) Train Algo.	Easy	Moderate	Hard	Easy	Moderate	Hard
(Clean Data)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	**77.79**	**67.69**	**66.31**	88.90	**85.64**	**78.86**
+TrainASN	71.74	61.78	60.26	87.29	77.08	75.89
+TrainSSN	75.54	66.26	63.72	88.07	79.18	78.03
+TrainSSNAlt	76.22	66.05	63.87	**89.00**	79.65	78.03
(Downsample ASN)	$AP_{3D}(\%)$			$AP_{BEV}(\%)$		
AVOD	36.13	27.39	26.39	77.60	59.84	51.82
+TrainASN	**71.30**	**56.04**	**49.08**	85.66	**70.17**	**68.55**
+TrainSSN	64.88	48.92	47.06	**86.21**	69.26	61.48
+TrainSSNAlt	48.98	36.30	31.06	75.00	51.35	49.60
(Downsample SSN)	$\min AP_{3D}(\%)$			$\min AP_{BEV}(\%)$		
AVOD	61.70	51.66	46.17	86.08	69.99	61.55
+TrainASN	65.74	53.49	51.35	82.27	67.88	65.79
+TrainSSN	**73.33**	**57.85**	**54.91**	**86.61**	**76.07**	**68.59**
+TrainSSNAlt	64.77	53.34	48.29	85.27	69.87	67.77
(Downsample SSN)	$\max \text{Diff} AP_{3D}(\%)$			$\max \text{Diff} AP_{BEV}(\%)$		
AVOD	11.71	5.88	3.59	1.96	7.60	8.65
+TrainASN	10.00	11.34	11.76	6.53	11.23	12.40
+TrainSSN	**0.94**	5.71	3.11	1.74	2.36	9.00
+TrainSSNAlt	6.98	**3.63**	**1.34**	**1.67**	**0.12**	**0.81**

Bibliography

Alireza Asvadi, Luís Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano J Nunes. Depthcn: Vehicle detection using 3d-lidar and convnet. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *European Conference on Computer Vision (ECCV) Workshops*, pages 613–627. Springer, 2014.

Massimo Bertozzi, Luca Bombini, Pietro Cerri, Paolo Medici, Pier Claudio Antonello, and Maurizio Miglietta. Obstacle detection and classification fusing radar and vision. In *Intelligent Vehicles Symposium (IV), 2008 IEEE*, pages 608–613. IEEE, 2008.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against

machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Markus Braun, Qing Rao, Yikang Wang, and Fabian Flohr. Pose-rcnn: Joint object detection and pose estimation using 3d object proposals. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 1546–1551. IEEE, 2016.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964, 2016.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, page 3. IEEE, 2017.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4774–4778, 2018.

Hyunggi Cho, Paul E Rybski, and Wende Zhang. Vision-based bicycle detection and tracking using a deformable part model and an ekf algorithm. In *Intelligent Transportation Systems (ITSC), 2010 IEEE 13th International Conference on*, pages 1875–1880, 2010.

Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1836–1843, 2014.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems (NeurIPS)*, pages 577–585, 2015.

Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3444–3453, 2017.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d recon-

structions of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, volume 1. IEEE, 2017.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems (NeurIPS)*, pages 379–387, 2016.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Conference on*, pages 886–893. IEEE, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, pages 248–255, 2009.

Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, pages 304–311, 2009.

Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

Xinxin Du, Marcelo H Ang, and Daniela Rus. Car detection for autonomous vehicle: Lidar and vision fusion approach through deep learning framework. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 749–754, 2017.

Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1355–1361. IEEE, 2017.

Markus Enzweiler and Darieu M Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009.

M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The pascal visual object classes challenge 2011 (voc 2011) results, 2010. URL http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1–8, 2008.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

Di Feng, Christian Haase-Schuetz, Lars Rosenbaum, Heinz Hertlein, Fabian Duffhauss, Claudius Glaeser, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous

driving: Datasets, methods, and challenges. *arXiv preprint arXiv:1902.07830*, 2019.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

David Geronimo, Antonio M Lopez, Angel D Sappa, and Thorsten Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (7):1239–1258, 2009.

Ross Girshick. Fast r-cnn. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1440–1448. IEEE, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

Alejandro González, Gabriel Villalonga, Jiaolong Xu, David Vázquez, Jaume Amores, and Antonio M López. Multiview random forest of local experts combining rgb and lidar data for pedestrian detection. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 356–361. IEEE, 2015.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International conference on learning representations (ICLR)*, 2015.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6645–6649, 2013.

Yanlei Gu and Shunsuke Kamijo. Recognition and pose estimation of urban road users from on-board camera for collision avoidance. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1266–1273, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361. Springer, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian

Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE signal processing magazine*, 29, 2012.

Jan Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele. Taking a deeper look at pedestrians. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4073–4082, 2015.

Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4700–4708, 2017.

Jing Huang and Brian Kingsbury. Audio-visual deep learning for noise robust speech recognition. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 7596–7599, 2013.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Jaekyum Kim, Junho Koh, Yecheol Kim, Jaehyung Choi, Youngbae Hwang, and Jun Won Choi. Robust deep multi-modal learning based on gated information fusion network. In *Asian conference on computer vision (ACCV)*, 2018a.

Taewan Kim and Joydeep Ghosh. Robust detection of non-motorized road users using deep learning on optical and lidar data. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 271–276. IEEE, 2016.

Taewan Kim, Michael Motro, Patrícia Lavieri, Saharsh Samir Oza, Joydeep Ghosh, and Chandra Bhat. Pedestrian detection with simplified depth prediction. In *Intelligent Transportation Systems (ITSC), 2018 IEEE 21th International Conference on*. IEEE, 2018b.

Ryan Kiros, Karteek Popuri, Dana Cobzas, and Martin Jagersand. Stacked multiscale feature learning for domain independent medical image segmentation. In *International workshop on machine learning in medical imaging*, pages 25–32. Springer, 2014.

Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–8, 2018.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 12697–12705, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436, 2015.

Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *European conference on computer vision (ECCV)*, pages 641–656, 2018.

Ming Liang, Bin Yang, Yun Chen, Rui Hui, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

Siqi Liu, Sidong Liu, Weidong Cai, Hangyu Che, Sonia Pujol, Ron Kikinis, Dagan Feng, Michael J Fulham, et al. Multimodal neuroimaging feature

learning for multiclass diagnosis of alzheimer's disease. *IEEE transactions on biomedical engineering*, 62(4):1132–1140, 2015.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Damien Matti, Hazım Kemal Ekenel, and Jean-Philippe Thiran. Combining lidar space clustering and convolutional neural networks for pedestrian detection. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–6. IEEE, 2017.

Oier Mees, Andreas Eitel, and Wolfram Burgard. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 151–156, 2016.

Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv preprint arXiv:1603.00831*, 2016.

Stefan Milch and Marc Behrens. Pedestrian detection with radar and

computer vision, 2001. URL http://www.smart-microwavesensors.de/Pedestrian_Detection.pdf.

Youssef Mroueh, Etienne Marcheret, and Vaibhava Goel. Deep multimodal learning for audio-visual speech recognition. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2130–2134, 2015.

Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

Cristiano Premebida, Joao Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining rgb and dense lidar data. In *IEEE International Conference on Intelligent Robots and Systems (IROS), 2014*, pages 4112–4117. IEEE, 2014.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 918–927, 2018.

Dhanesh Ramachandram and Graham W Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE signal processing magazine*, 34(6):96–108, 2017.

Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6517–6525. IEEE, 2017.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

Payam Sabzmeydani and Greg Mori. Detecting pedestrians by learning shapelet features. In *Computer Vision and Pattern Recognition (CVPR), 2007 IEEE Conference on*, pages 1–8. IEEE, 2007.

Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 8614–8618, 2013.

Khaled Saleh, Mohammed Hossny, Ahmed Hossny, and Saeid Nahavandi. Cyclist detection in lidar scans using faster r-cnn and synthetic depth images. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–6. IEEE, 2017.

Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. Regnet:

Multimodal sensor registration using deep neural networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1803–1810. IEEE, 2017.

Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 618–626, 2017.

Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–779, 2019.

Martin Simonovsky, Benjamín Gutiérrez-Becker, Diana Mateus, Nassir Navab, and Nikos Komodakis. A deep metric for multimodal registration. In *International conference on medical image computing and computer-assisted intervention*, pages 10–18. Springer, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Chao Sui, Mohammed Bennamoun, and Roberto Togneri. Listening with your eyes: Towards a practical visual speech recognition system using deep

boltzmann machines. In *IEEE international conference on computer vision (ICCV)*, pages 154–162, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–9, 2015.

Tomoji Takasu. Rtklib: An open source program package for gnss positioning, 2013. URL http://www.rtklib.com/.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International conference on learning representations (ICLR)*, 2019.

Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996a.

Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996b.

Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

Abhinav Valada, Johan Vertens, Ankit Dhall, and Wolfram Burgard. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In *IEEE international conference on robotics and automation (ICRA)*, pages 4644–4651, 2017.

Tao Wang, Nanning Zheng, Jingmin Xin, and Zheng Ma. Integrating millimeter wave radar with a monocular vision sensor for on-road obstacle detection applications. *Sensors*, 11(9):8992–9008, 2011.

Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *IEEE intelligent vehicles symposium (IV)*, pages 1–6, 2018.

Bo Wu and Ram Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Computer Vision (ICCV), 2005 IEEE International Conference on*, volume 1, pages 90–97. IEEE, 2005.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. Online multimodal deep similarity learning with application to image retrieval. In *21st ACM international conference on multimedia*, pages 153–162. ACM, 2013.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929, 2016.

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint arXiv:1711.06396*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.

Vita

Taewan Kim received B.S. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2012, and M.S.E. degree in Electrical Engineering-Systems from the University of Michigan, Ann Arbor in 2014. In the summer of 2018, he worked as an applied scientist intern at Amazon.com, Inc. His doctoral research at the University of Texas at Austin has been advised by Dr. Joydeep Ghosh. His primary research interests are in machine learning, data mining, and deep learning focusing on utilizing multiple input sources, with applications to computer vision and robotics.

Permanent address: twankim@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.